Teach-Discover-Treat 2014, Part 3 ; 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 antiparasitic protein target, calcium-dependent protein kinase 1 (CDPK1), is used as a target for molecular docking. Molecular docking is described, using DOCK3.7 as a system for virtual screening. Crystal structures of CDPK1 in *Cryptosporidium parvum* and *Toxoplasma gondii* are validated as virtual screening targets and suitable for docking, as is a homology model of CDPK1 belonging to the parasite *Eimeria tenella*. Relative predicted affinity values for a held-out test set of 22 analogs of known ligand scaffolds were 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 lead-like molecules from the purchasable eMolecules database.

http://www.tdtproject.org/challenge-3---cdpk1-virtual-screening.html

-1. Philosophy and Background

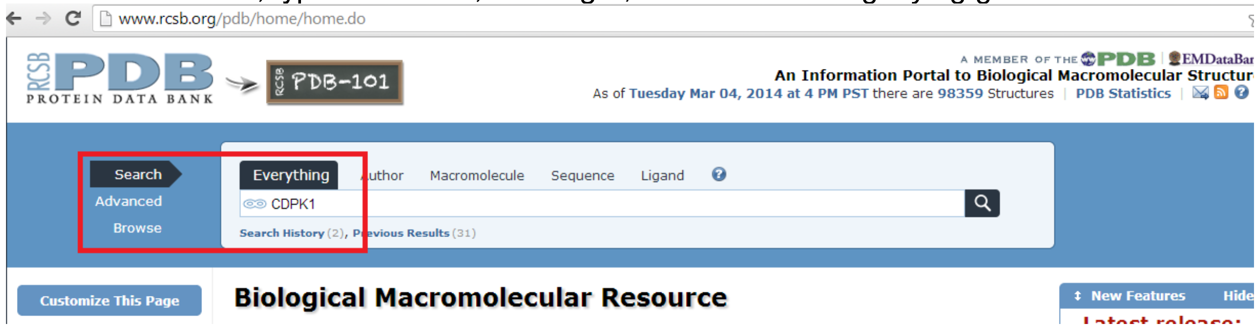
Molecular Docking is the process of using a protein structure (either experimentally determined or modeled) to evaluate the position and energy of small molecules in the binding site of the structure. Many approximations must be made in consideration of computer time and resources, but a good program will use a reasonable energy function and consider many thousand poses of each small molecule relative to the protein binding site. At best, the resulting energy score of the predicted protein-ligand complexes is a relative enthalpy and not a true binding affinity, since the scoring function neglects physical terms for protein and ligand internal energy, the hydrophobic effect of the protein, and receptor desolvation, among others. However, docking can be used to predict likely ligand binding poses in the absence of a co-crystal structure, propose possible ligand modifications to improve affinity, or, its main strength, to discover novel scaffolds that fit in the same binding pocket for new chemical leads or probes. 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 utilizing pre-built small molecule databases, so the actual docking and scoring 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, such as PyMOL and UCSF Chimera.

In this project, we will be docking to structures (both x-ray crystallographic and homology modeled) of an enzyme called calcium-dependent kinase 1 (CDPK1). This enzyme is present in a variety of clinically-relevant human parasites, including *Cryptosporidium parvum* (causes cryptosporidiosis), *Toxoplasma gondii* (causes toxoplasmosis), *Eimeria tenella* (causes coccidosis), and *Plasmodium falciparum* (causes malaria).

Cryptosporidosis and Toxoplasmosis infections and symptoms can be tempered by the human immune system, however this is not the case in immunocompromised patients. Coccidosis is a major disease affecting poultry, with no effective treatment options. Finally, malaria, an enormously prevalent disease throughout the world, has many treatment options, but resistance has been reported against all known classes of small molecule drugs.

Previously, CDPK1 was determined to be a secondary target of two promiscuous inhibitors of a different *T. gondii* kinase, cGMP-dependent protein kinase (PKG). A kinase is an enzyme that transfers phosphate groups from a donor molecule, such as ATP, to another substrate protein. As a result, most kinase inhibitors work by binding in the same site as ATP, competitively displacing ATP and making the kinase unable to phosphorylate any substrates or function normally. To see how these work, let's compare a crystal structure of CDPK1 with ATP bound and one with an inhibitor bound. Go to the Protein Data Bank (PDB), a public repository for experimentally determined crystal structures of proteins (http://www.rcsb.org/pdb/home/home.do).

In the 'Search' box, type in CDPK1, our target, and click the magnifying glass to search:



An Information Portal to Biological Macromolecular Structur

As of Tuesday Mar 04, 2014 at 4 PM PST there are 98359 Structures | PDB Statistics

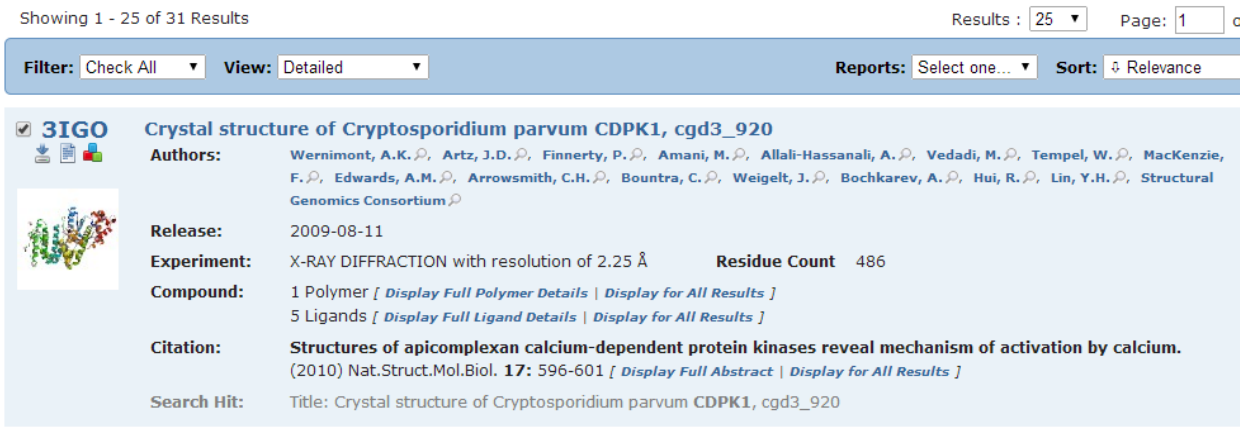
CDPK1

Search History (2), Previous Results (31)

Biological Macromolecular Resource

Latest release:

You'll get several results, but let's investigate the first one, from *C. parvum*, with identifier 3IGO (all PDB codes are 4 digits):



Showing 1 - 25 of 31 Results

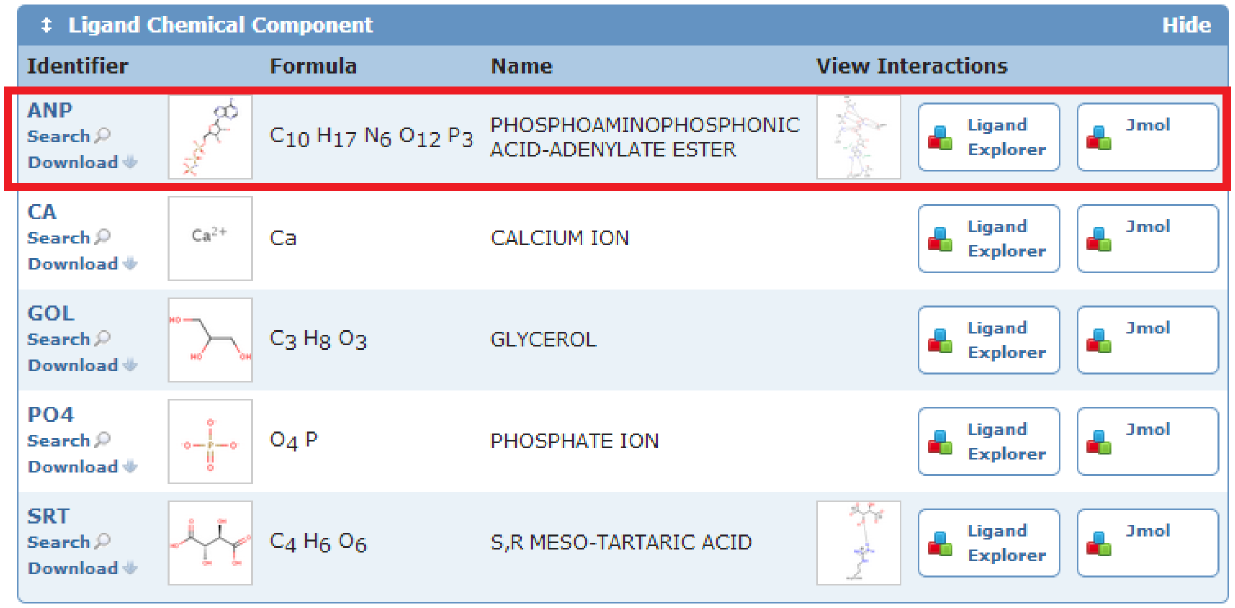
Results : 25 Page: 1

Filter: Check All View: Detailed Reports: Select one... Sort: Relevance

3IGO Crystal structure of Cryptosporidium parvum CDPK1, cgd3_920

Authors:	Wernimont, A.K., Artz, J.D., Finnerty, P., Amani, M., Allali-Hassanali, A., Vedadi, M., Tempel, W., MacKenzie, F., Edwards, A.M., Arrowsmith, C.H., Bountra, C., Weigelt, J., Bochkarev, A., Hui, R., Lin, Y.H., Structural Genomics Consortium
Release:	2009-08-11
Experiment:	X-RAY DIFFRACTION with resolution of 2.25 Å Residue Count 486
Compound:	1 Polymer [Display Full Polymer Details | Display for All Results] 5 Ligands [Display Full Ligand Details | Display for All Results]
Citation:	**Structures of apicomplexan calcium-dependent protein kinases reveal mechanism of activation by calcium.** (2010) Nat.Struct.Mol.Biol. **17:** 596-601 [Display Full Abstract | Display for All Results]
Search Hit:	Title: Crystal structure of Cryptosporidium parvum **CDPK1**, cgd3_920

Upon clicking on 3IGO, you'll come to a page with a lot of information on the structure, such as the published journal article the information is associated with, the sequence of the protein, crystallographic parameters, etc. Scroll down until you see the co-crystallized small molecule information ('Ligand Chemical Component'):

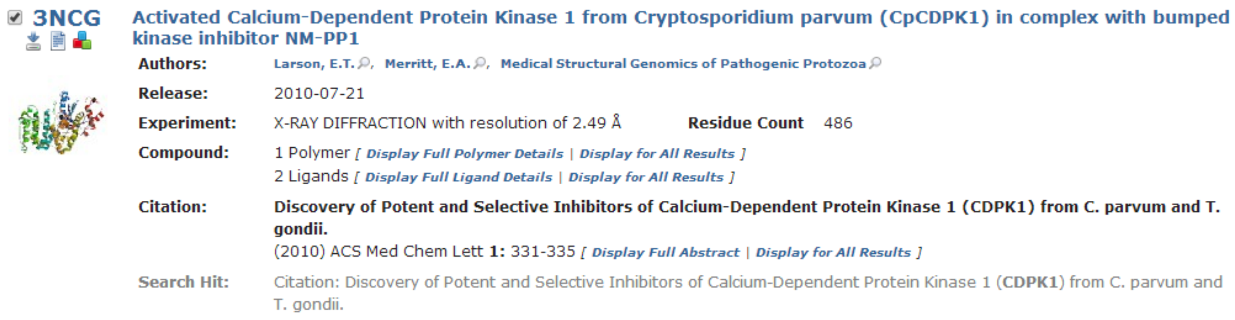


Ligand Chemical Component Hide

Identifier		Formula	Name	View Interactions		
ANP Search Download		$C_{10}H_{17}N_6O_{12}P_3$	PHOSPHOAMINOPHOSPHONIC ACID-ADENYLATE ESTER		Ligand Explorer	Jmol
CA Search Download	Ca^{2+}	Ca	CALCIUM ION		Ligand Explorer	Jmol
GOL Search Download		$C_3H_8O_3$	GLYCEROL		Ligand Explorer	Jmol
PO4 Search Download		O_4P	PHOSPHATE ION		Ligand Explorer	Jmol
SRT Search Download		$C_4H_6O_6$	S,R MESO-TARTARIC ACID		Ligand Explorer	Jmol

You can see that the first row is an ATP analogue, ANP. So, this crystal structure of CDPK1 was co-crystallized with an ATP analogue in the binding site – perfect!

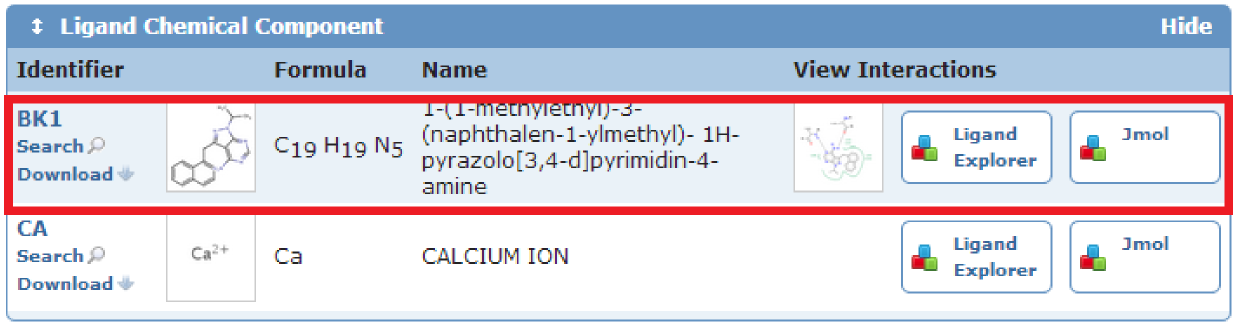
Next, let's find a crystal structure of CDPK1 form the same species with an inhibitor bound. Going back to the search results, scroll down until you get to 3NCG, another CDPK1 crystal structure from a parasite we're interested in, *C. parvum*



3NCG **Activated Calcium-Dependent Protein Kinase 1 from Cryptosporidium parvum (CpCDPK1) in complex with bumped kinase inhibitor NM-PP1**

Authors:	Larson, E.T., Merritt, E.A., Medical Structural Genomics of Pathogenic Protozoa
Release:	2010-07-21
Experiment:	X-RAY DIFFRACTION with resolution of 2.49 Å **Residue Count** 486
Compound:	1 Polymer [*Display Full Polymer Details* | *Display for All Results*] 2 Ligands [*Display Full Ligand Details* | *Display for All Results*]
Citation:	**Discovery of Potent and Selective Inhibitors of Calcium-Dependent Protein Kinase 1 (CDPK1) from C. parvum and T. gondii.** (2010) ACS Med Chem Lett **1:** 331-335 [*Display Full Abstract* | *Display for All Results*]
Search Hit:	Citation: Discovery of Potent and Selective Inhibitors of Calcium-Dependent Protein Kinase 1 (**CDPK1**) from C. parvum and T. gondii.

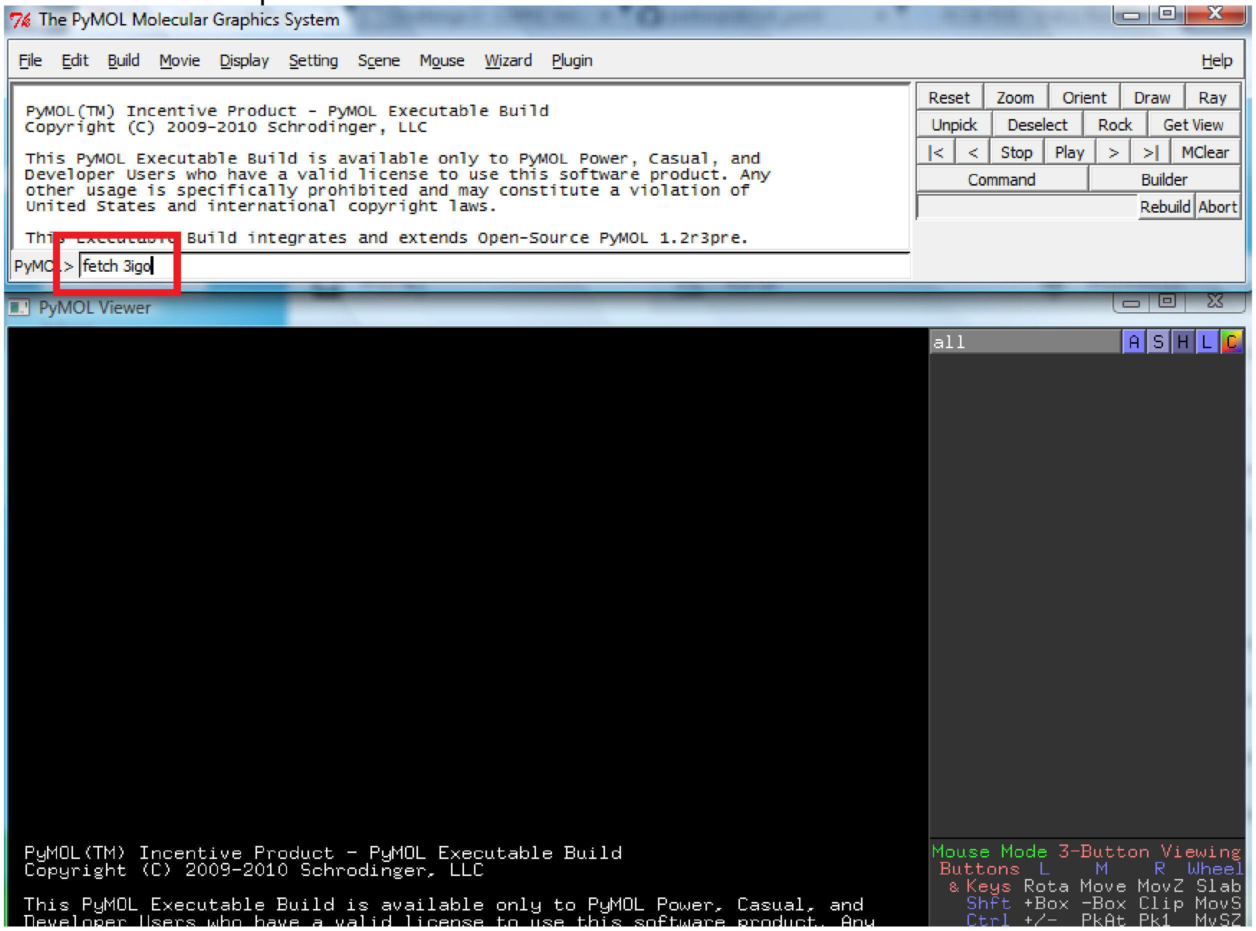
Its ligand information reveals it was co-crystallized with the inhibitor BK1:



Ligand Chemical Component Hide

Identifier		Formula	Name	View Interactions		
BK1 Search Download		$C_{19}H_{19}N_5$	1-(1-methylethyl)-3-(naphthalen-1-ylmethyl)- 1H-pyrazolo[3,4-d]pyrimidin-4-amine		Ligand Explorer	Jmol
CA Search Download	Ca^{2+}	Ca	CALCIUM ION		Ligand Explorer	Jmol

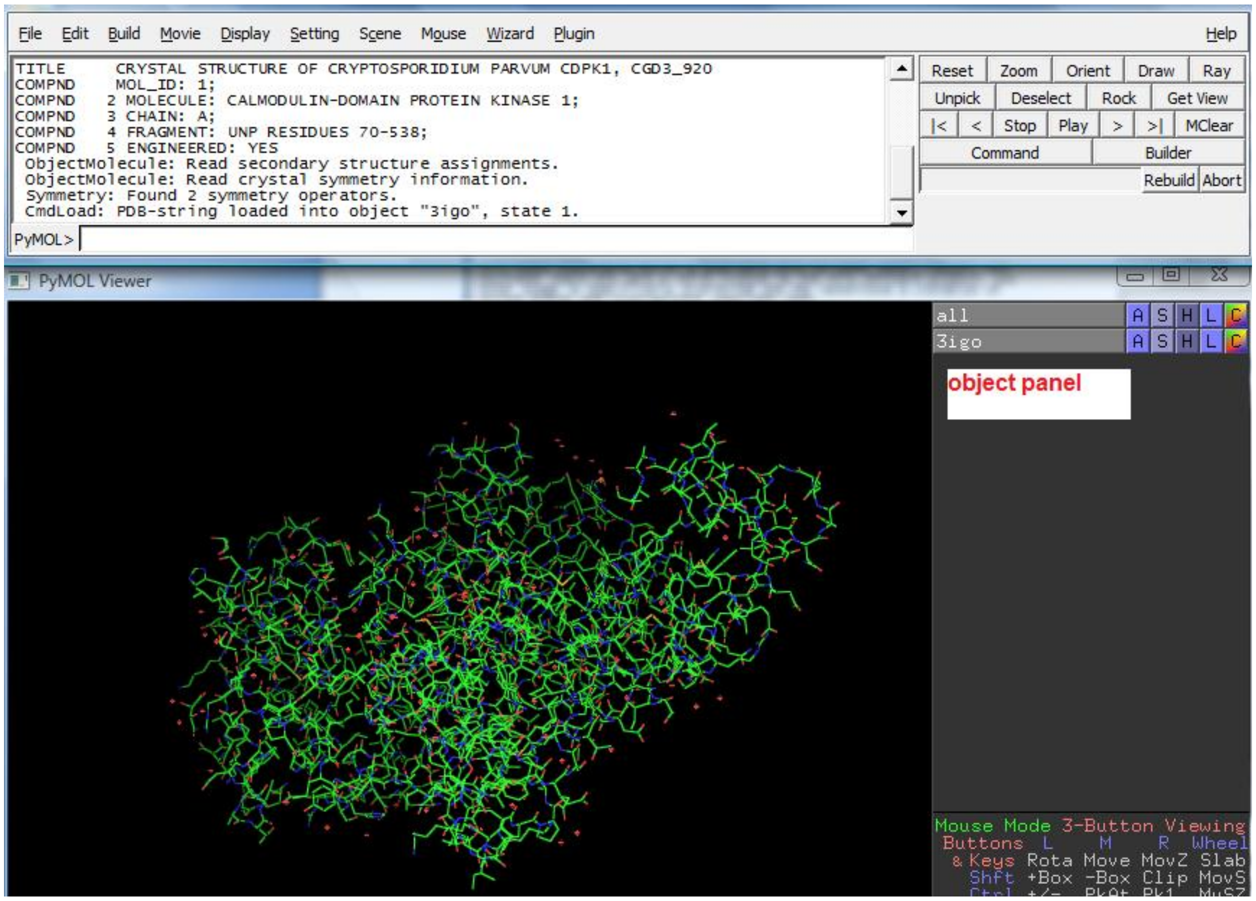
Now, we can load these structures in PyMOL, a protein visualization software. This can be downloaded for free at http://pymol.org/educational/, and installed on the appropriate platform according to instructions at http://pymol.org/install. After opening up PyMOL, type the command ‘fetch 3igo’ (the PDB code of the ATP-bound CDPK1 structure) in the command line and press Enter:


The PyMOL Molecular Graphics System
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Reset Zoom Orient Draw Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Command Builder
Rebuild Abort
PyMOL Viewer
all A S H L C
Mouse Mode 3-Button Viewing
Buttons L M R Wheel
& Keys Rota Move MovZ Slab
Shft +Box -Box Clip MovS
Ctrl +/- PkAt Pk1 MvSZ

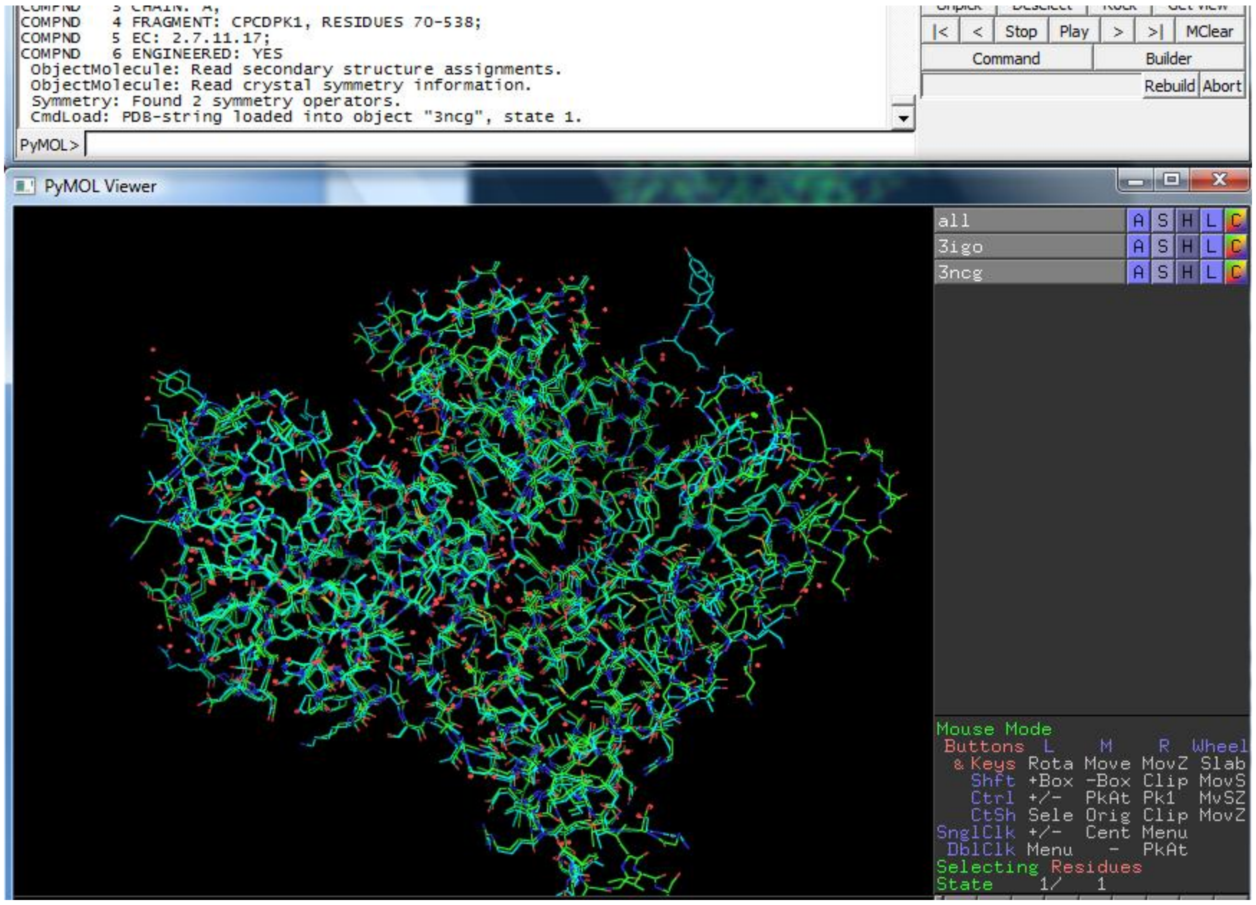
```
PyMOL(TM) Incentive Product - PyMOL Executable Build
Copyright (C) 2009-2010 Schrodinger, LLC

This PyMOL Executable Build is available only to PyMOL Power, Casual, and
Developer Users who have a valid license to use this software product. Any
other usage is specifically prohibited and may constitute a violation of
United States and international copyright laws.

This Executable Build integrates and extends Open-Source PyMOL 1.2r3pre.
PyMOL> fetch 3igo
```

```
PyMOL(TM) Incentive Product - PyMOL Executable Build
Copyright (C) 2009-2010 Schrodinger, LLC

This PyMOL Executable Build is available only to PyMOL Power, Casual, and
Developer Users who have a valid license to use this software product. Any
```

This will automatically download the structure of 3igo saved as 3igo.pdb in whichever directory you loaded PyMOL, as well as show the structure of it on the screen (with carbons colored green, oxygens red, sulfurs yellow, and nitrogens blue), saved as the ‘object’ 3igo on the object panel on the right:


```
TITLE     CRYSTAL STRUCTURE OF CRYPTOSPORIDIUM PARVUM CDPK1, CGD3_920
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: CALMODULIN-DOMAIN PROTEIN KINASE 1;
COMPND   3 CHAIN: A;
COMPND   4 FRAGMENT: UNP RESIDUES 70-538;
COMPND   5 ENGINEERED: YES
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 2 symmetry operators.
 CmdLoad: PDB-string loaded into object "3igo", state 1.
PyMOL>
```
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Reset Zoom Orient Draw Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Command Builder
Rebuild Abort
PyMOL Viewer
all A S H L C
3igo A S H L C
object panel
Mouse Mode 3-Button Viewing
Buttons L M R Wheel
& Keys Rota Move MovZ Slab
Shft +Box -Box Clip MovS

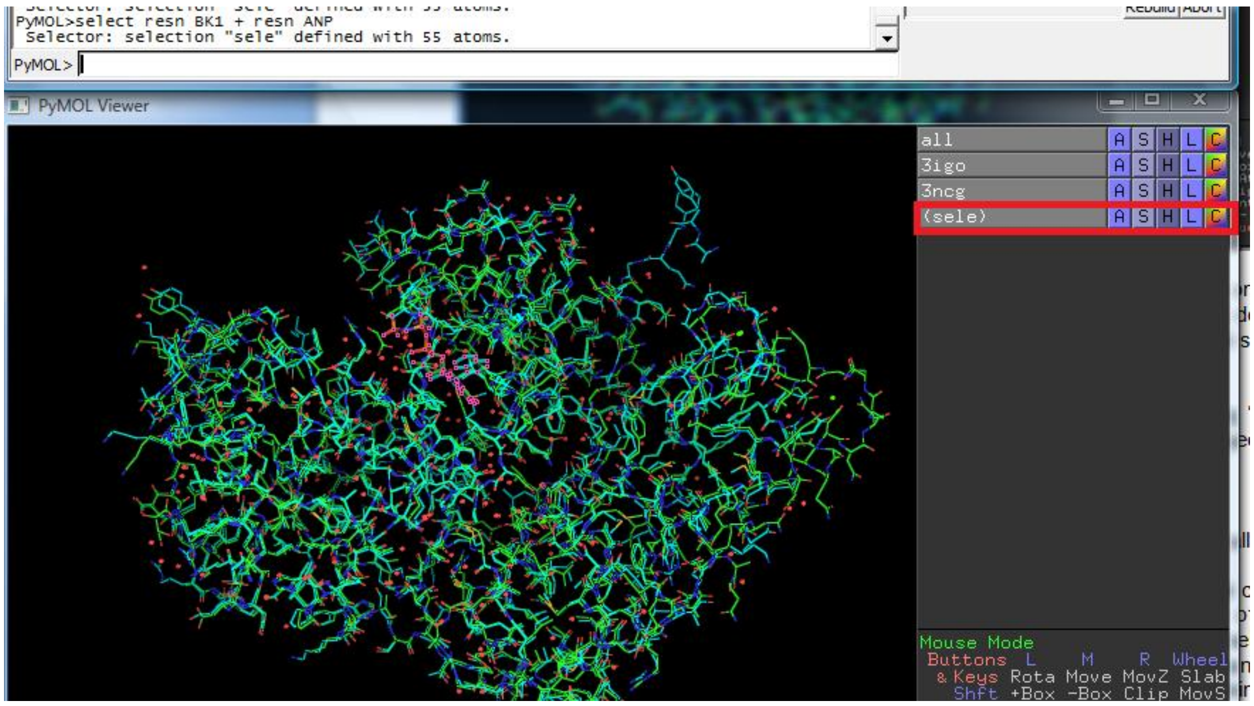
Now let’s also load the structure of CDPK1 with an inhibitor bound by entering ‘fetch 3ncg.’ 3ncg will have carbons colored cyan, oxygens red, sulfurs yellow, and nitrogens blue, to differentiate it easily from the 3igo structure.



```
COMPND    4 FRAGMENT: CPCDPK1, RESIDUES 70-538;
COMPND    5 EC: 2.7.11.17;
COMPND    6 ENGINEERED: YES
 ObjectMolecule: Read secondary structure assignments.
 ObjectMolecule: Read crystal symmetry information.
 Symmetry: Found 2 symmetry operators.
 CmdLoad: PDB-string loaded into object "3ncg", state 1.
PyMOL>
```

Now, even though these are structures of the same kinase, and they look almost on top of each other, we need to align the structures to see how ANP and BK1 overlap. To do this, simply enter ‘align 3igo, 3ncg.’ Since they started out very close, there is minimal shifting of the structures to overlap.

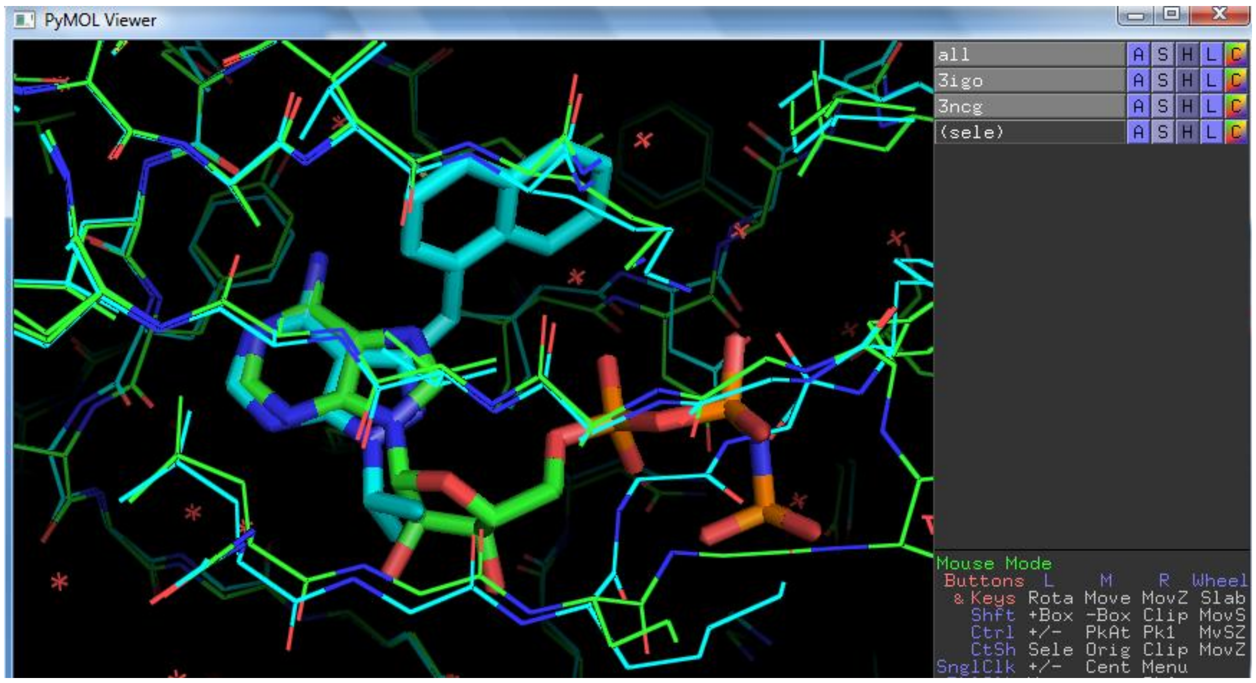
We want to see the bound small molecules, so let’s select BK1 and ANP by typing ‘select resn BK1 and ANP.’ This selects the molecules BK1 and ANP and creates an object called (sele) on the object panel.



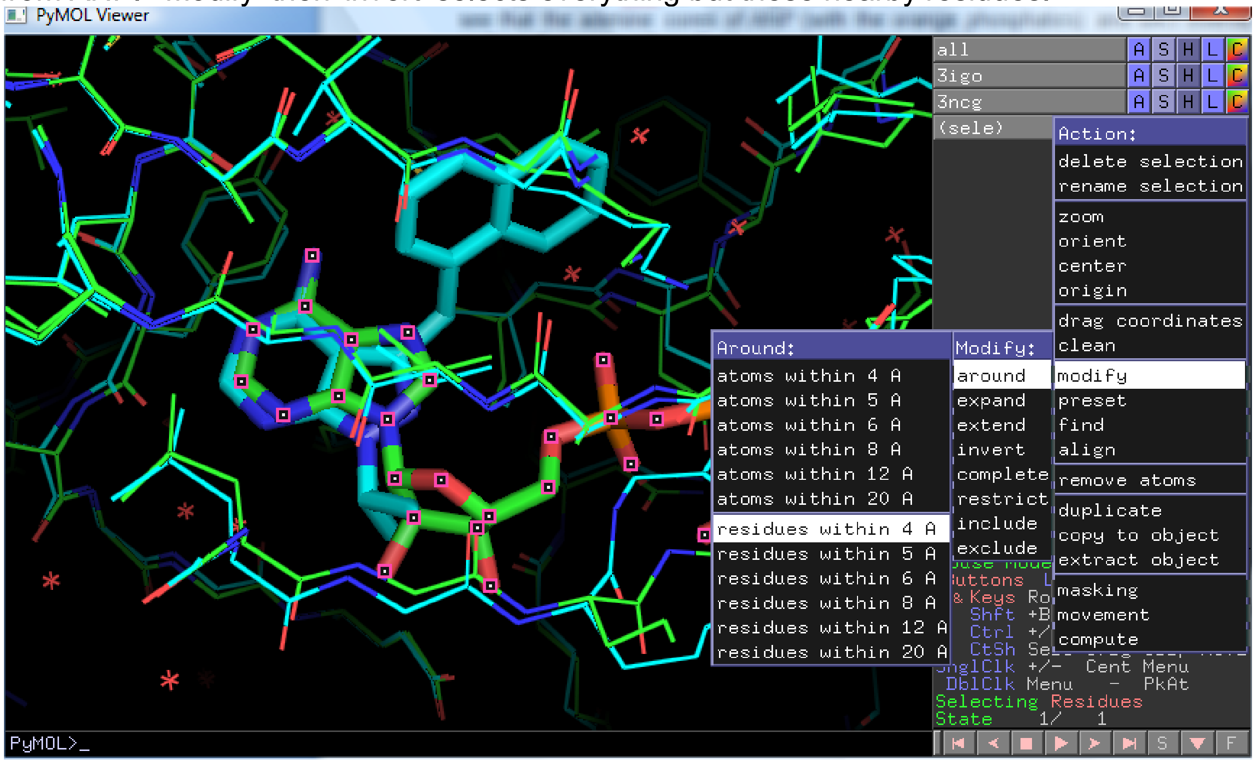
```
PyMOL>select resn BK1 + resn ANP
 Selector: selection "sele" defined with 55 atoms.
PyMOL>
```

Type ‘show sticks, sele’ to represent our selection, BK1 and ANP, as thick sticks, instead of thin default wires.

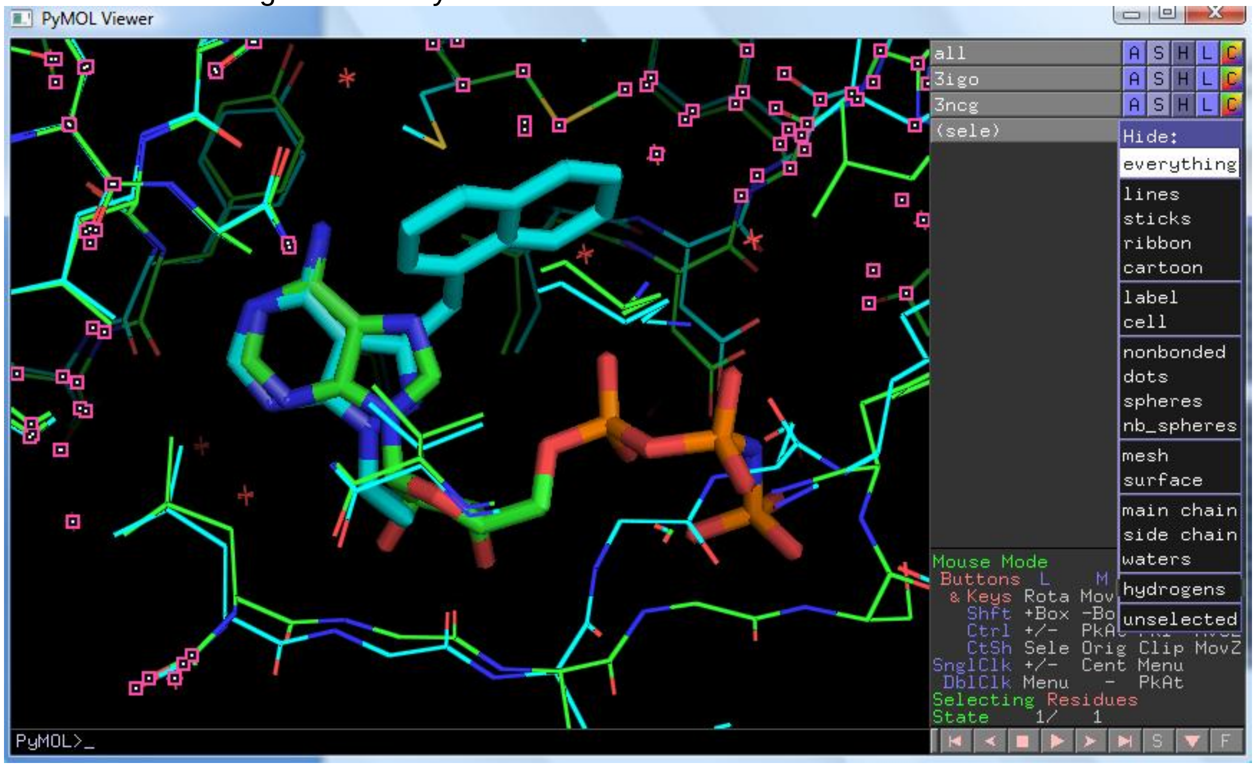
Now type ‘zoom sele’ to focus right to where BK1 and ANP are in the structure. You can see that the adenine cores of ANP (with the orange phosphates) and BK1 overlap perfectly, making the same interactions.



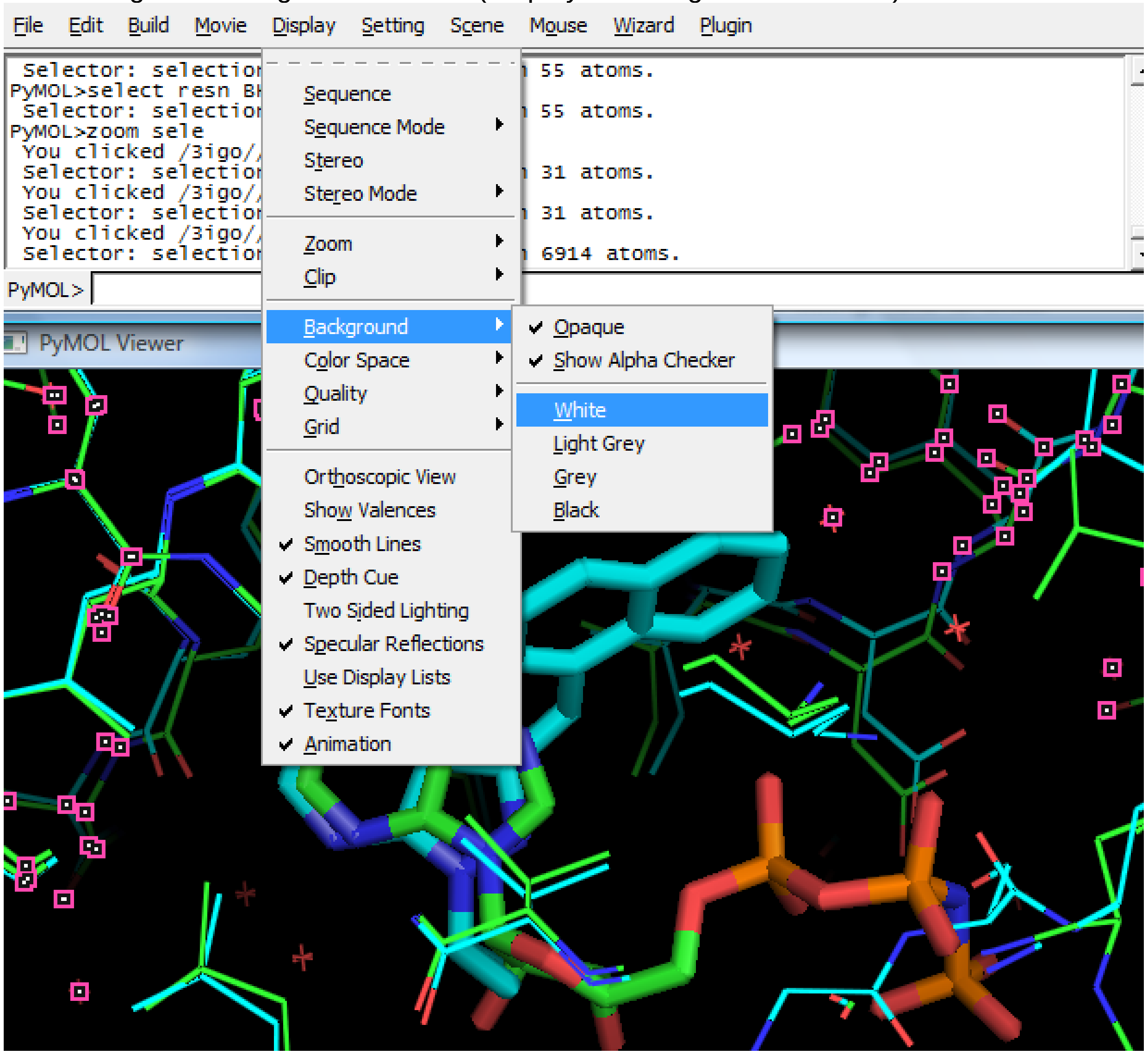
If we click on ANP to select, and then under the A (Action) menu for ‘sele’ in the object panel, ‘modify,’ then ‘around,’ we can look at residues that are only a certain distance away from ANP. ‘Modify’ then 'invert' selects everything but these nearby residues.



Then, under the H (hide) menu for ‘sele,’ select everything. This hides every residue but those that are 5 Angstroms away from ANP.

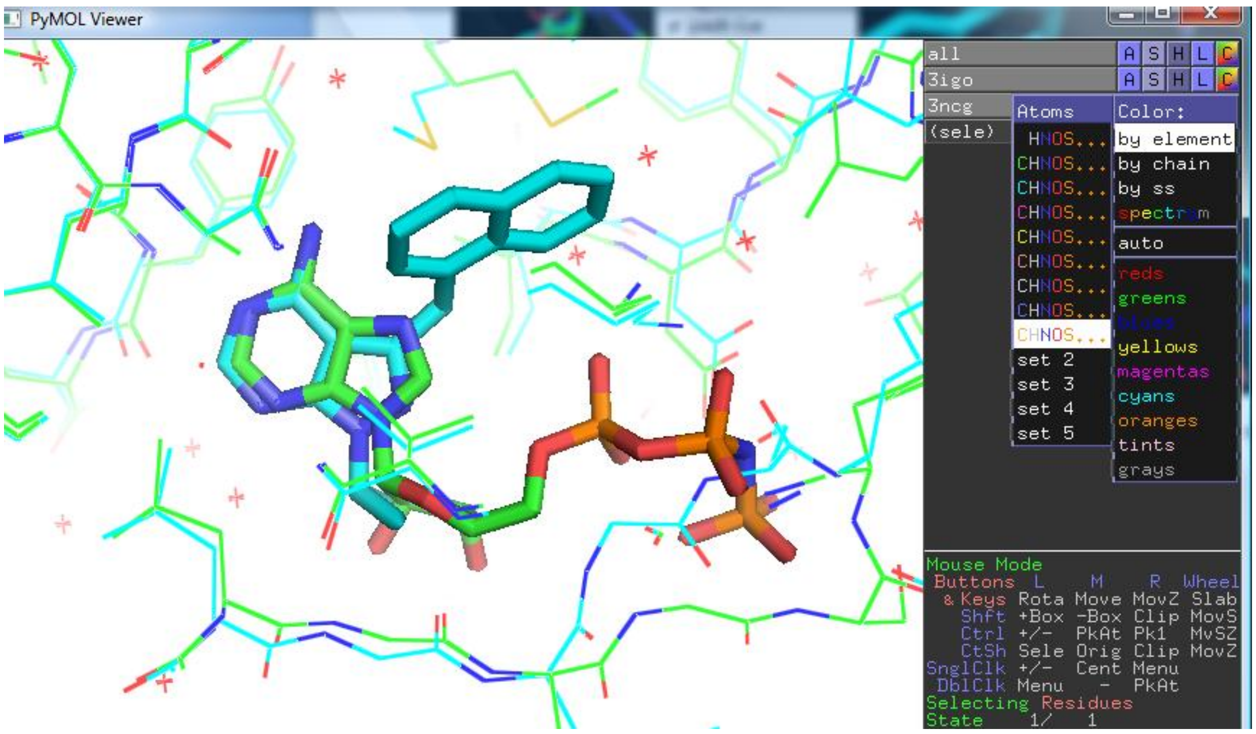


If we change the background to white (Display → Background → White):

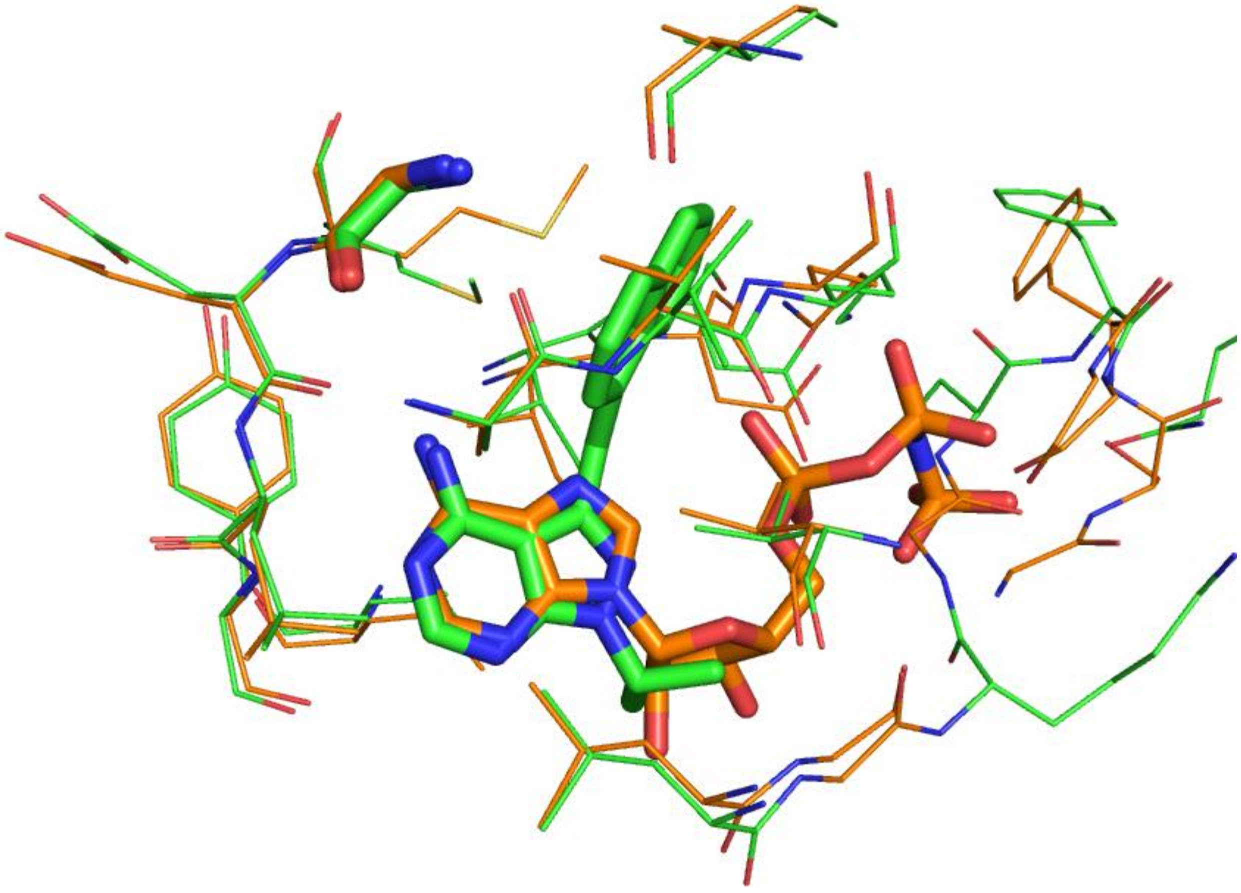


```
 Selector: selectio                       55 atoms.
PyMOL>select resn B
 Selector: selectio                       55 atoms.
PyMOL>zoom sele
 You clicked /3igo/
 Selector: selectio                       31 atoms.
 You clicked /3igo/
 Selector: selectio                       31 atoms.
 You clicked /3igo/
 Selector: selectio                       6914 atoms.
```

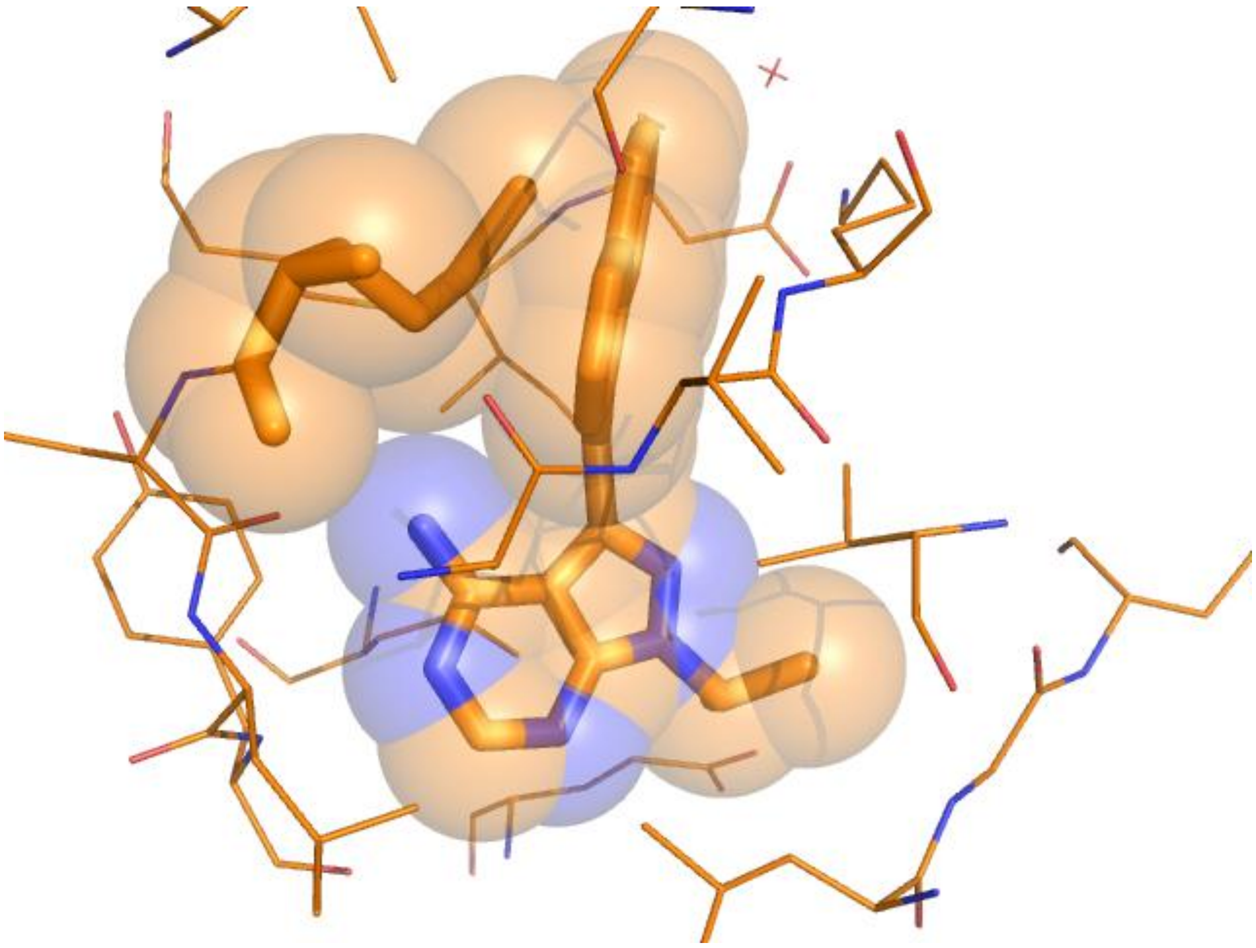
And the color of carbons in 3ncg to orange (the C for colors menu → by element → orange carbons):



We can now easily the two ligands and protein differences:



Highlighted in sticks is Gly152, which is also the residue position nicknamed the 'gatekeeper' residue in kinases. No human kinases have a glycine gatekeeper residue. If this residue were bigger, such as a methionine residue like is common in human kinases, the bulky naphthyl substituent would sterically clash with the BK1 ligand:



This class of kinase inhibitors, which have bulky substituents coming off of the adenine ring, are called "bumped" inhibitors, since they are too big to fit into human kinase binding sites. Humans also have many kinases, and one strategy to inhibit parasitic kinases specifically is to target this kinase with bumped inhibitors. *T. gondii* also has a glycine gatekeeper residue, and, so, is also inhibited by this class of molecules. However, the target *E. tenella* and *P. falciparum* have threonine gatekeeper residues, which have slightly bigger side-chains. Only a small percentage of human kinases have a Thr gatekeeper, of which, src is one. This substitution may have an effect on inhibitor binding and human vs. parasitic selectivity, which is what we will be examining in this challenge. We will use homology modeling to model the structures of *E. tenella* CDPK1 and molecular docking to hypothesize the structures of a held out test set of 22 molecules bound to CDPK1 from various species. Finally, we will dock purchasable molecules from eMolecules to the binding site and identify additional molecules with novel scaffolds that could be tested experimentally. These molecules could have enhanced binding affinities, specificity, or other novel properties that will make them suitable as lead molecules to treat these parasitic infections.

0. Downloading & installing software.

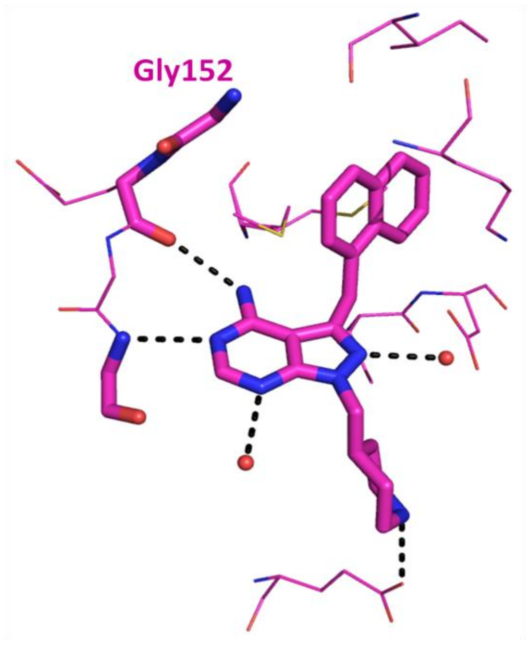
DOCK3.7 runs on many flavors of GNU/Linux, as do all the accessory tools. Some popular versions include CentOS, Ubuntu and Debian. GNU/Linux is the most commonly used operating systems on large clusters of computers, so hopefully you have access to a small cluster already, but it is possible to install 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 (except for the industry license which costs money). 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 & available online). Extra scripts for the TDT 2014 Challenge are located in https://github.com/ryancoleman/tdt2014-part2/ and https://github.com/joelkarpiak/tdt_part3, which 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. Also, UCSF Chimera is available here: http://www.cgl.ucsf.edu/chimera/download.html

1. Preparation of crystal structures 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 do to multiple scientists naming proteins before a standard was decided upon). Since multiple co-crystal structures of CDPK1 from *T. gondii* and *C. parvum* exist, let's download many of them and compare. Since human src also has a Thr gatekeeper, and crystal structures of the protein exist with a bumped kinase inhibitor (BKI), let's look at those, as well. After searching in the PDB for CDPK1 structures from both species, open PyMOL and fetch the following codes: 3igo, 3ncg, 3mwu (for *C. parvum*); 3uqg, 3uqf (for human Src); 3i79, 4mxa, 4mx9, 4,97, 4m84, 3v5t, 3v5p, 3v51, 3upz, 3upx, 3t3v, 3t3u, 3sxf, 3sx9, 3i7b, 3i7c, 3n51 (for *T. gondii*). We can structurally align all of them by selecting one (here, arbitrarily, 3V51), and aligning as earlier. This is included as xtals_align.pse

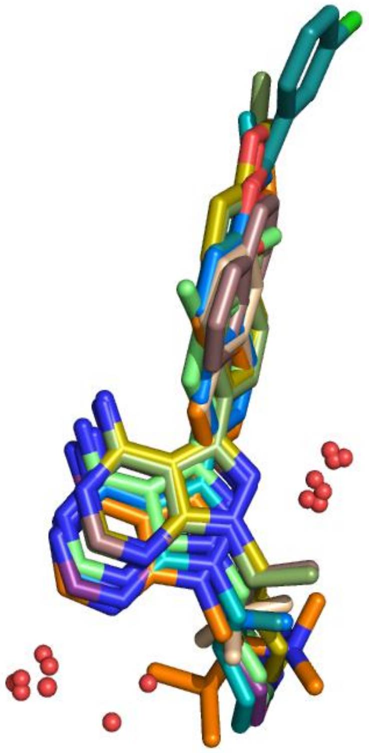
There are a few important observations:

1) There are conserved interactions across species. Comparing *C. parvum* (3MWU) to any of the *T. gondii* structures shows the same conserved interactions between the protein and bound ligand. Here, black dashed lines represent polar hydrogen bonding interactions, while the residues highlighted in sticks (the 'hinge region') are making interactions with the adenine ring of the bumped kinase inhibitor. Also in sticks is the glycine gatekeeper residue:

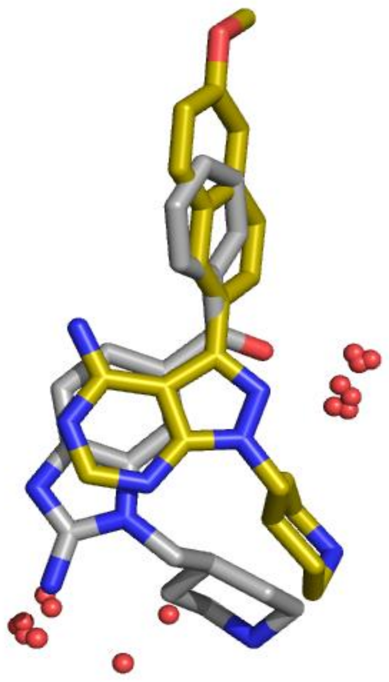


C. parvum

2) There are conserved water interactions across species. In the higher-resolution *T. gondii* structures that show waters, the same ones exist to make interactions with the ligand as in *C. parvum*:

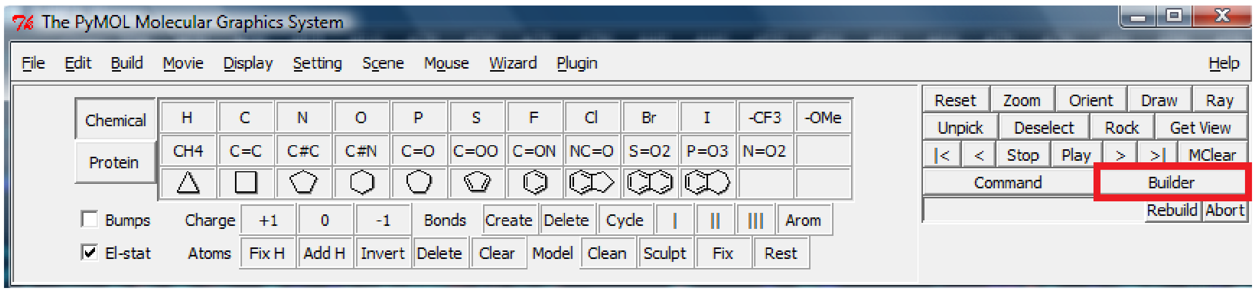


3) One water is conserved across chemotypes. *T. gondii* was crystallized with another chemotype lacking an adenine ring (3upx) that shows a conserved water making a similar interaction as with a BKI. The lower left water, however, would be displaced by the binding of this new scaffold:

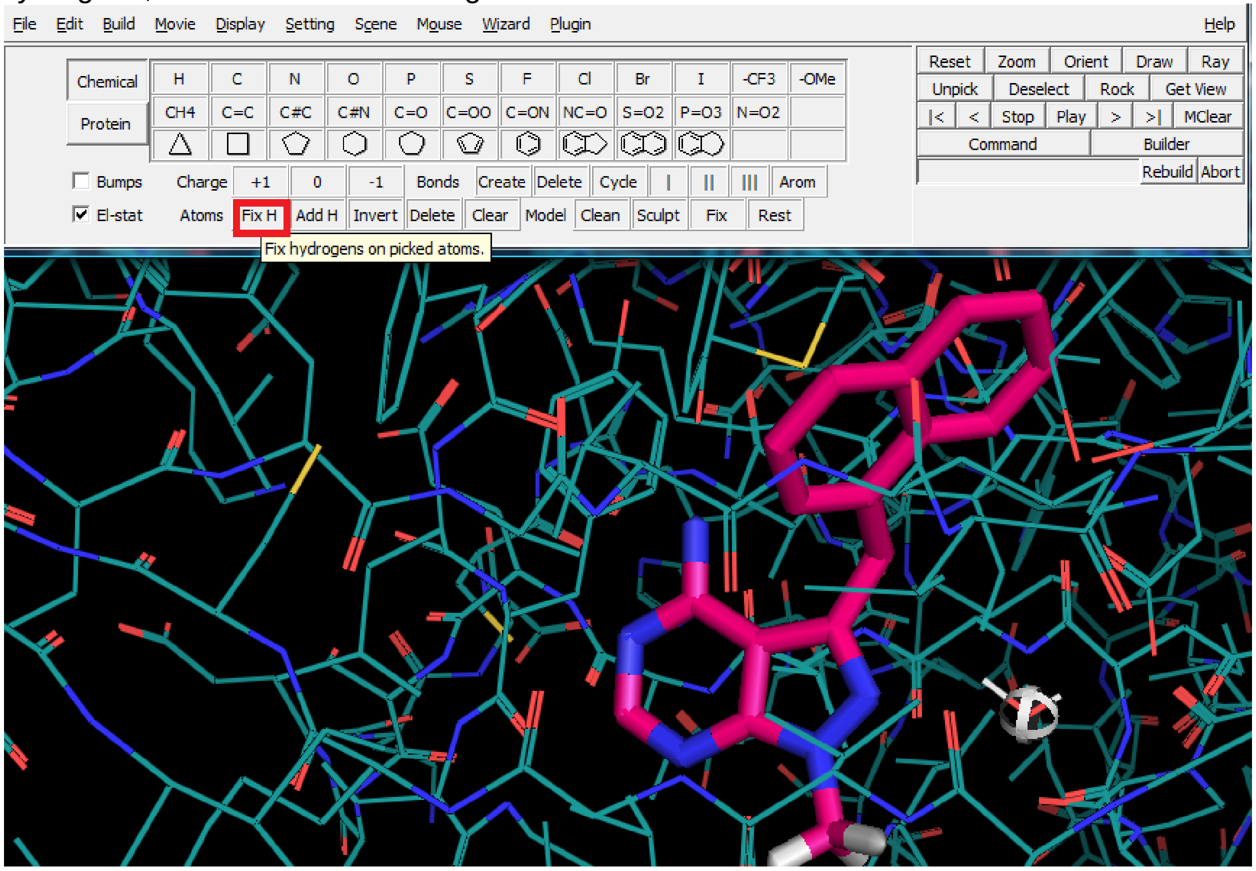


For docking, we are only going to use a single crystal structure from each species. For *C. parvum,* we'll choose a structure with: the conserved waters, an extended ligand reaching past the gatekeeper with a bulky naphthyl group, and one with an extended ring substituent off of the adenine moiety that ion-pairs with Glu159. This is 3mwu. All of these characteristics will help place ligands during the docking. For *T. gondii,* we'll choose a structure with a conserved water, as well as one with a very extended substituent past the gatekeeper residue. This is 3v51.
.

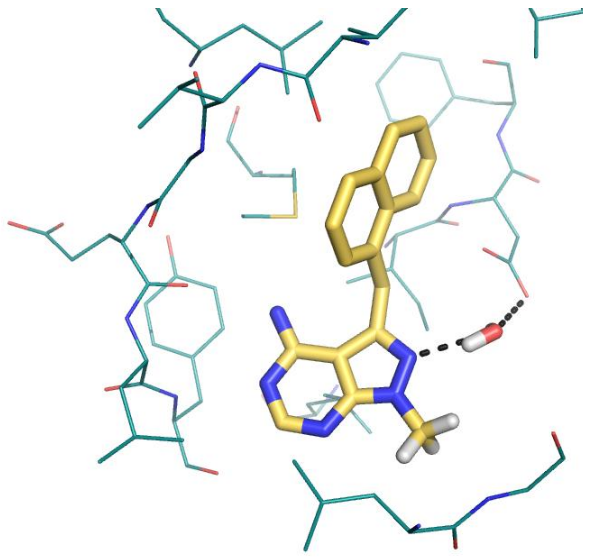
Next, we'll need to manually position the hydrogens in the conserved water, HOH residue 630 in 3mwu. First, go into the Builder module of PyMOL:


The PyMOL Molecular Graphics System
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
Chemical
Protein
H C N O P S F Cl Br I -CF3 -OMe
CH4 C=C C#C C#N C=O C=OO C=ON NC=O S=O2 P=O3 N=O2
Bumps Charge +1 0 -1 Bonds Create Delete Cycle | || ||| Arom
El-stat Atoms Fix H Add H Invert Delete Clear Model Clean Sculpt Fix Rest
Reset Zoom Orient Draw Ray
Unpick Deselect Rock Get View
|< < Stop Play > >| MClear
Command Builder
Rebuild Abort

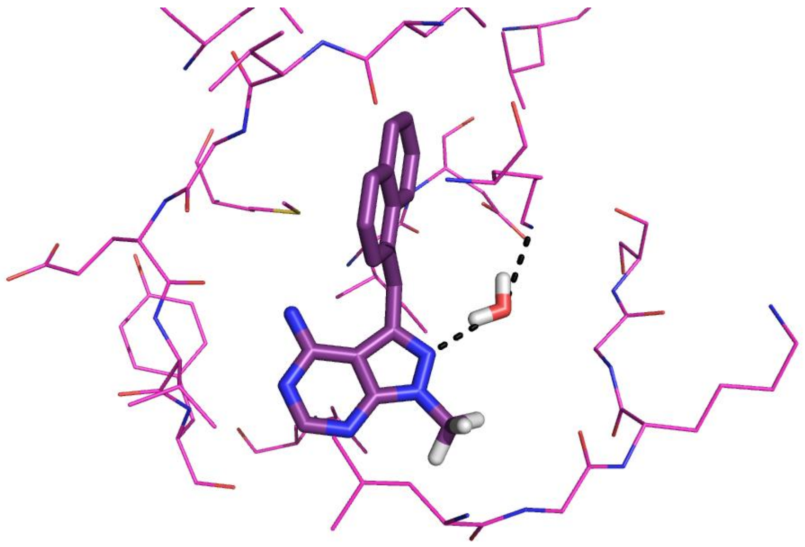
Then, select the water, and add hydrogens to it by clicking ‘Fix H.’ This will arbitrarily add hydrogens, albeit at the correct angles.



Finally, since these hydrogens do not create the correct hydrogen bonding pattern, select one of the hydrogens and hold Control + Shift + Left Click to move each hydrogen so that the O position remains fixed. Point the hydrogens towards Asp259 and the adenine ring of the ligand, to look like:



Perform the same operations on 3V51 for *T. gondii* (HOH 508) and point the hydrogens towards Lys80 and the adenine ring of the ligand. Asp195, the equivalent of Asp259 in 3mwu, is present, but it's ion-pair Lys80 is closer to the water in this structure.



Finally, we will need to separate files for the co-crystallized ligand (to serve as the starting-point for docking) and the receptor-conserved water. Everything else needs to be stripped away. To save the ligand, type 'select xtal-lig, resn I76' for 3V51 and 'select xtal-lig, resn BK3'. This creates an object of just the ligand, called xtal-lig. To save this, type 'save xtal-lig.pdb, xtal-lig'. This saves the ligand as a file called xtal-lig.pdb

Then, we need to change HETATM in the pdb to ATOM records. To do this, type the Linux command (the backticks ` are not part of the command):

```
`cat xtal-lig.pdb | sed 's/HETATM/ATOM  /g' > xtal-lig.pdb`
```

These files can be found at:
https://github.com/joelkarpiak/tdt_part3/Cpar_3mwu_xtal_docking_onewater/xtal-lig.pdb and
https://github.com/joelkarpiak/tdt_part3/Tgon_3v51_xtal_docking_onewater/xtal-lig.pdb

To select just the protein and water 508 that we manually changed, first type ‘select resn hoh and !(resi 508)’ followed by ‘extract waters, sele’. This will take out all of the waters but 508 and place them in a separate object. Repeat for the ligand: ‘select resn I76’ and ‘extract ligand, sele.’ Finally, ‘save rec.pdb, 3V51’. This creates a file called rec.pdb that is just what is leftover in the 3V51 object, which is the protein and water 508. Then, we just have to repeat the commands for 3mwu, replacing water 508 with water 630.

In rec.pdb, however, the water is still listed as a HETATM, so we have to change this to an ATOM record by issuing the command:

`cat rec.pdb | sed 's/HETATM/ATOM  /g' > rec.pdb`

These files can be found at:
https://github.com/joelkarpiak/tdt_part3/Cpar_3mwu_xtal_docking_onewater/rec.pdb and
https://github.com/joelkarpiak/tdt_part3/Tgon_3v51_xtal_docking_onewater/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 (grids of pre-computed energy functions, as well as mapping the location of the binding site) are a bit large, they are available here:
https://github.com/joelkarpiak/tdt_part3/Tgon_3v51_xtal_docking_onewater/working
https://github.com/joelkarpiak/tdt_part3/Tgon_3v51_xtal_docking_onewater/dockfiles
https://github.com/joelkarpiak/tdt_part3/Cpar_3mwu_xtal_docking_onewater/working
https://github.com/joelkarpiak/tdt_part3/Cpar_3mwu_xtal_docking_onewater/dockfiles

What went on when you ran blastermaster? This can be seen in the blaster.log.txt file. The grand philosophy is that the binding site is being identified by the ligand, and the energy grids for each of the three energy functions (van der Waals, ligand desolvation and electrostatics) are being pre-computed and constructed. Once these grids are constructed, docking proceeds very quickly since each atom can be scored against the static grids for each energy function. While using grids is just an approximation, it does make DOCK3.7 very fast.

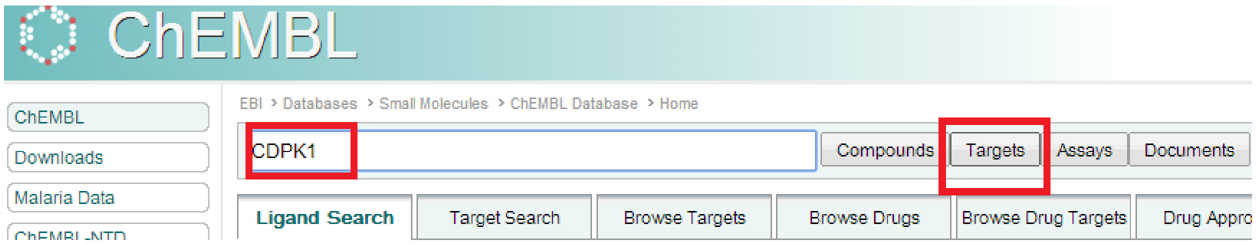
Once you've got these files, you're ready to build ligands and then move on to docking.

Preparing the Ligands

Literature Compounds

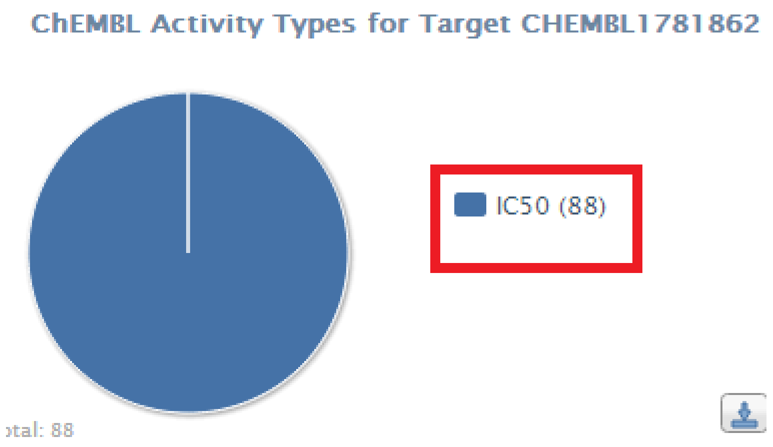
First, we need to identify ligands of these proteins. Then, to build them for docking, we need to start with a SMILES string to represent the atom connectivity.

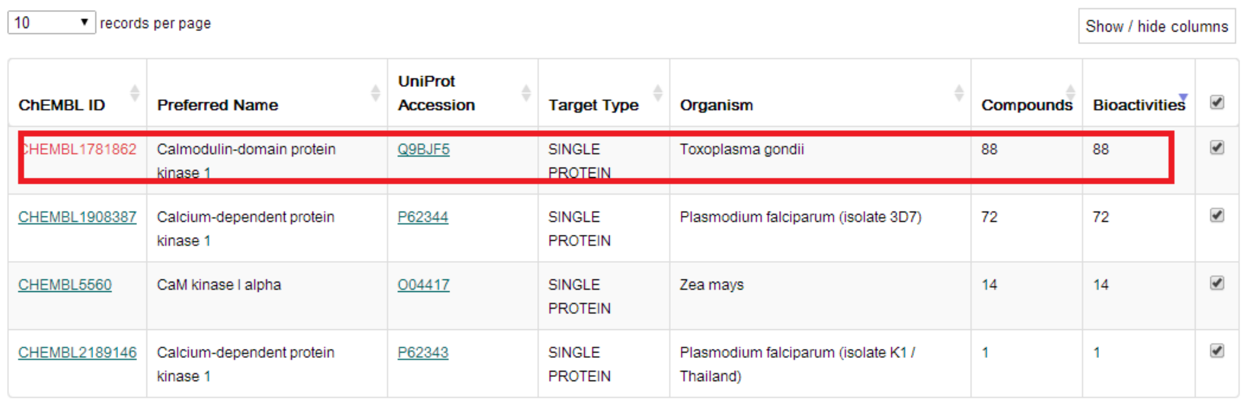
To find these, let's to go to ChEMBL, a public repository of protein-ligand binding data from the literature (https://www.ebi.ac.uk/chembl/). There, type in CDPK1, and click 'Target search,' since this is the protein for which we want to find ligands:



One of our targets, CDPK1 for *T. gondii,* is the first result, with 88 bioactivities. This means there are 88 data points (IC_{50}s, Kis, or some other measure) for compounds binding to *T. gondii* CDPK1.

IC_{50} is the only data type available, so select that:



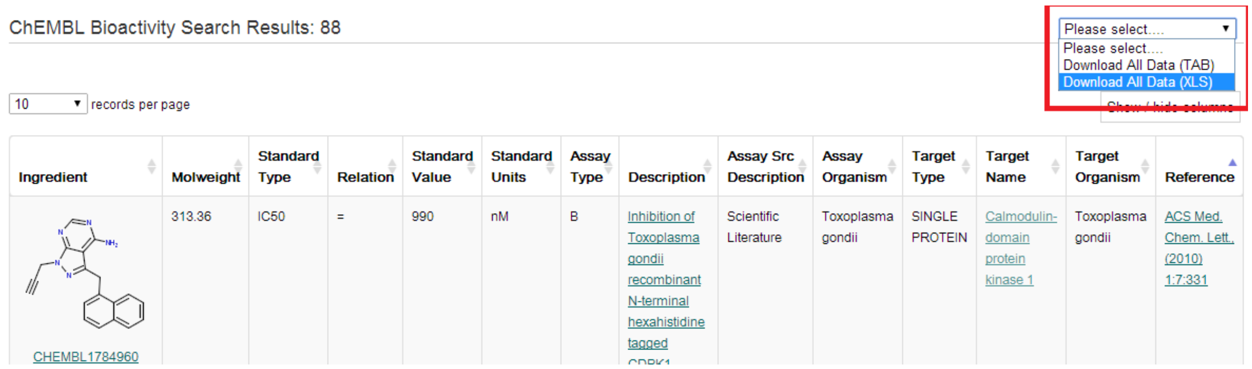


10 records per page

Show / hide columns

ChEMBL ID	Preferred Name	UniProt Accession	Target Type	Organism	Compounds	Bioactivities	
CHEMBL1781862	Calmodulin-domain protein kinase 1	Q9BJF5	SINGLE PROTEIN	Toxoplasma gondii	88	88	✓
CHEMBL1908387	Calcium-dependent protein kinase 1	P62344	SINGLE PROTEIN	Plasmodium falciparum (isolate 3D7)	72	72	✓
CHEMBL5560	CaM kinase I alpha	O04417	SINGLE PROTEIN	Zea mays	14	14	✓
CHEMBL2189146	Calcium-dependent protein kinase 1	P62343	SINGLE PROTEIN	Plasmodium falciparum (isolate K1 / Thailand)	1	1	✓

Next, select to download the data as an XLS file:

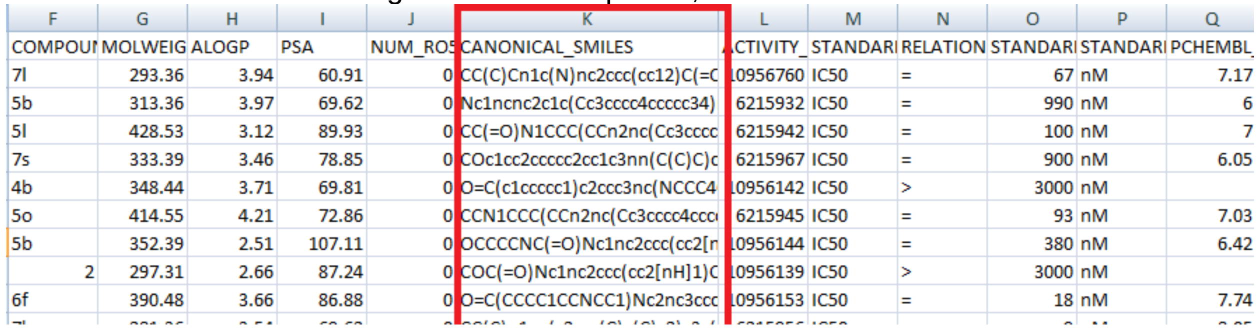


ChEMBL Bioactivity Search Results: 88

10 records per page

Ingredient	Molweight	Standard Type	Relation	Standard Value	Standard Units	Assay Type	Description	Assay Src Description	Assay Organism	Target Type	Target Name	Target Organism	Reference
CHEMBL1784960	313.36	IC50	=	990	nM	B	Inhibition of Toxoplasma gondii recombinant N-terminal hexahistidine tagged	Scientific Literature	Toxoplasma gondii	SINGLE PROTEIN	Calmodulin-domain protein kinase 1	Toxoplasma gondii	ACS Med. Chem. Lett. (2010) 1:7:331

The result is an extensive spreadsheet (bioactivities.xls) of chemical properties, however, all we need is the SMILES strings for each compound, column K:



F	G	H	I	J	K	L	M	N	O	P	Q
COMPOUI	MOLWEIG	ALOGP	PSA	NUM_ROS	CANONICAL_SMILES	CTIVITY_	STANDARI	RELATION	STANDARI	STANDARI	PCHEMBL_
7l	293.36	3.94	60.91	0	CC(C)Cn1c(N)nc2ccc(cc12)C(=C	10956760	IC50	=	67	nM	7.17
5b	313.36	3.97	69.62	0	Nc1ncnc2c1c(Cc3cccc4ccccc34)	6215932	IC50	=	990	nM	6
5l	428.53	3.12	89.93	0	CC(=O)N1CCC(CCn2nc(Cc3cccc	6215942	IC50	=	100	nM	7
7s	333.39	3.46	78.85	0	COc1cc2ccccc2cc1c3nn(C(C)C)c	6215967	IC50	=	900	nM	6.05
4b	348.44	3.71	69.81	0	O=C(c1ccccc1)c2ccc3nc(NCCC4	10956142	IC50	>	3000	nM	
5o	414.55	4.21	72.86	0	CCN1CCC(CCn2nc(Cc3cccc4ccc	6215945	IC50	=	93	nM	7.03
5b	352.39	2.51	107.11	0	OCCCCNC(=O)Nc1nc2ccc(cc2[n	10956144	IC50	=	380	nM	6.42
2	297.31	2.66	87.24	0	COC(=O)Nc1nc2ccc(cc2[nH]1)C	10956139	IC50	>	3000	nM	
6f	390.48	3.66	86.88	0	O=C(CCCC1CCNCC1)Nc2nc3ccc	10956153	IC50	=	18	nM	7.74

Also, we need an identifier for the docking program to name each compound. You can use either column A, the ChEMBL identifier, or any other column. We chose column F, the chemical identifier used in the paper the data originates from; this is helpful in case you want to go look up the compounds in the paper. Since these identifiers are not unique, we will need to add something to the beginning of each of these identifiers. We chose the journal of the paper: acsmed and bioorg. To create an input file for ligand database generation, we need a first column of SMILES strings, followed by a space, then the identifier.

To generate these files, you can run the following commands:

‘grep ACS bioactivities | awk –F “\t” ‘{print $11” “”acsmed”$6}’ >& acsmed.smi
‘grep Bio bioactivities | awk –F “\t” ‘{print $11” “”bioorg”$6’ >& bioorg.smi
‘cat acsmed.smi bioorg.smi >& lit.smi’

This file can be found at https://github.com/joelkarpiak/tdt_part3/lit_build/lit.smi

Once you have the input file, you can build a docking database with the downloaded scripts from https://github.com/ryancoleman/tdt2014-part2/tree/master/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 lit.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`

This 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 ‘lit’ makes sense. Once that is done you should have a file called cladosporin-0000001.db2.gz Again, feel free to compare this file to those in https://github.com/joelkarpiak/tdt_part3/lit_build/.

Test Set Compounds

The IC_{50} data from 22 compounds has been withheld from participants in this challenge, so our goal is to predict which of the compounds may bind to the *T. gondii*, *C. parvum*, and *E. tenella* kinases.

Eventually, the file we want to start with is here:
http://www.tdtproject.org/uploads/6/8/2/0/6820495/tdt2-challenge3-cdpk1_externaltestset.txt

This can be turned into our .smi format by running the command:

‘awk ‘{print $2” “$1}’ tdt2-challenge3-cdpk1_externaltestset.txt >& 22ligs.smi’, and then removing the top row of column names.

This file can be checked against this file here: https://github.com/joelkarpiak/tdt_part3/22ligs_build/

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

`db2start.e.cxcalc.sh 22ligs.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 22ligs`

The hard part (building the ligand database) is over and now the docking is relatively fast and easy.

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/joelkarpiak/tdt_part3/22ligs_build/

Running DOCK

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, docking is easy! First, for both *T. gondii* and *C. parvum* builds, set up the docking run with the following command (part of the DOCK3.7 download):

`setup_db2.py /path/to/lit_database/`

Then run it with

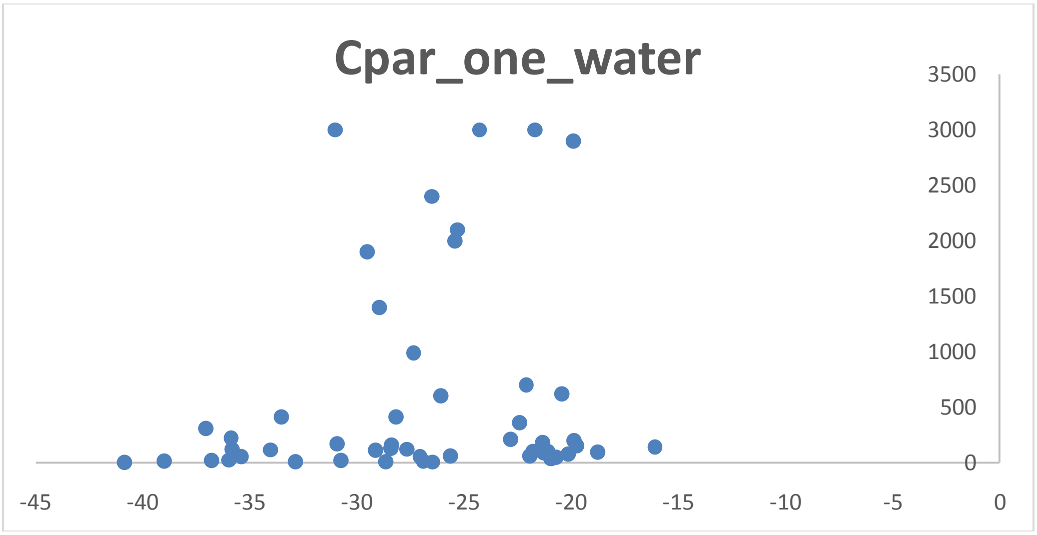
`submit.csh`

Once the run is done, you can run the following 2 programs to get a file with the best pose for each ligand in the lit database. DOCK3.7 came up with hundreds of thousands of possible poses for each and scored them all.

`extract_all.py`
`getposes.py -o Cpar_3mwu_xtal_docking_onewater.mol2` or ‘-o Tgon_3v51_xtal_docing_onewater.mol2’

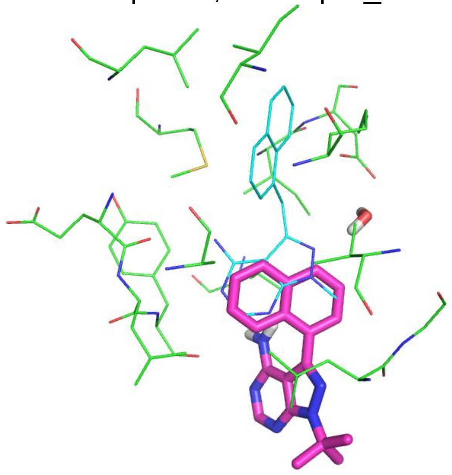
These can be compared to the versions here:
https://github.com/joelkarpiak/tdt_part3/Tgon_3v51_xtal_docking_onewater/ and
https://github.com/joelkarpiak/tdt_part3/Cpar_3mwu_xtal_docking_onewater/

Since these compounds are a training set to basically see if docking can relatively score these compounds correctly and get their poses right (before we try docking the 22 compound held-out test set). Looking at the file extract_all.sort.uniq.txt, the Total Score column gives the final docked energy score for the best-scoring pose of each ligand. Using the IC_{50}s from the ChEMBL data spreadsheet, we can compare the relative order of the experimentally determined IC_{50}s to the docked energy score. Although dock scores are not equivalent to binding affinities, scores of a close series of SAR could correlate to the relative ranking of those compounds by IC_{50}s. More negative docking scores (x-axis) should be equivalent to lower IC_{50}s (y-axis). However, we see that for both kinases, docked scores do not correlate with IC_{50}s:

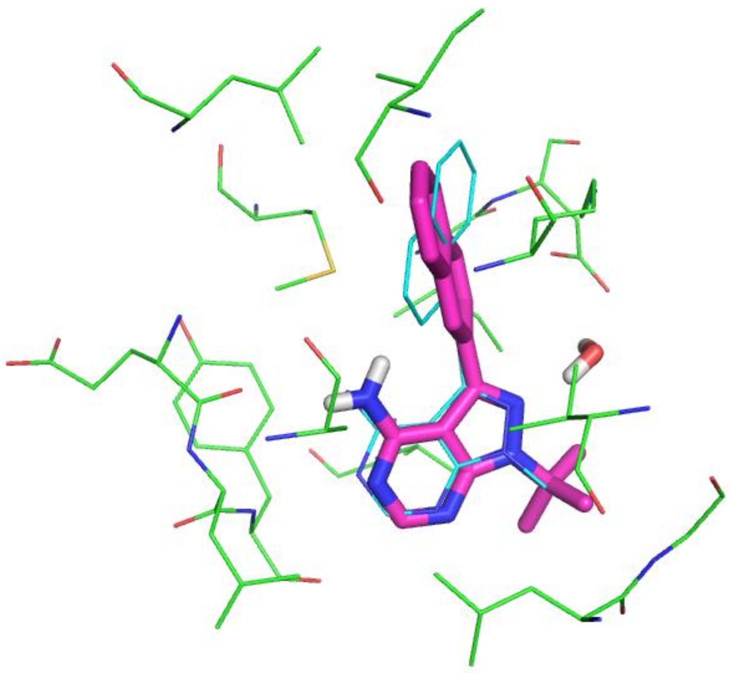


All graphs can be found in https://github.com/joelkarpiak/tdt_part3/training_set_data.xls

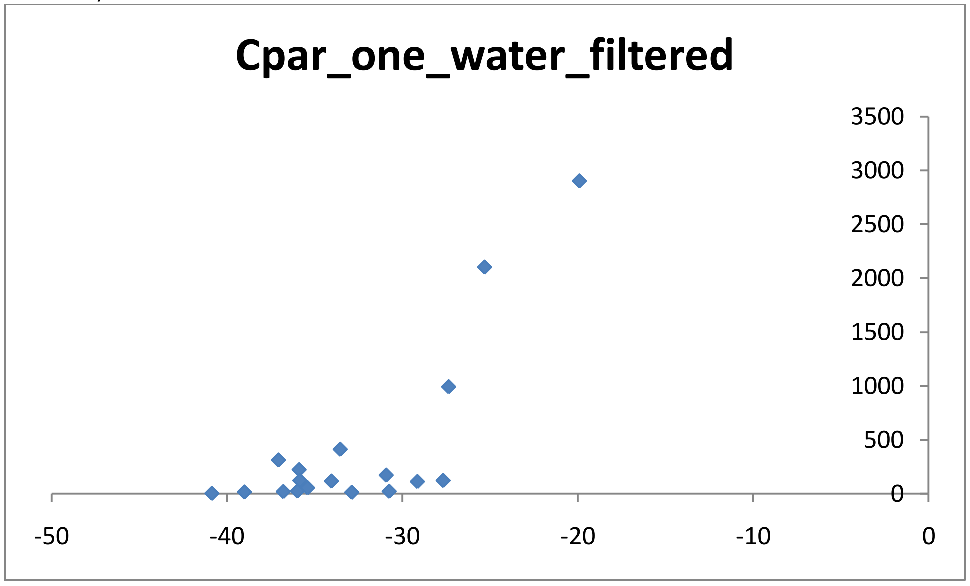
If the scores don't correlate with IC_{50}s, is this due to the scoring or is the actual pose wrong? To look at poses, load Cpar_3mwu_xtal_docking_onewater.mol2 in PyMOL. Some ligands,



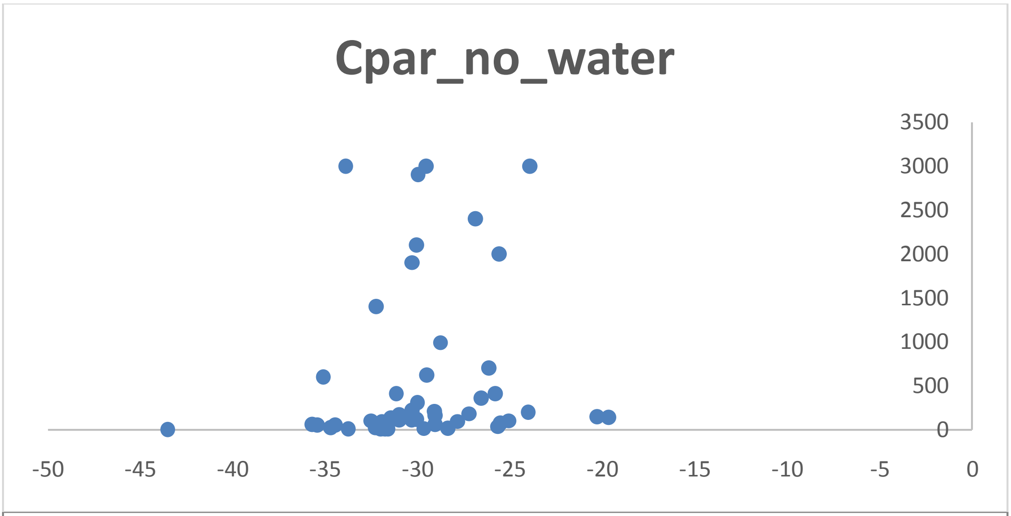
such as acsmedchem1 (shown in magenta, over the cyan xtal-lig.pdb used as a starting point), are clearly wrong. Others, such as acsmedchem2 (again, in magenta), are clearly correct.

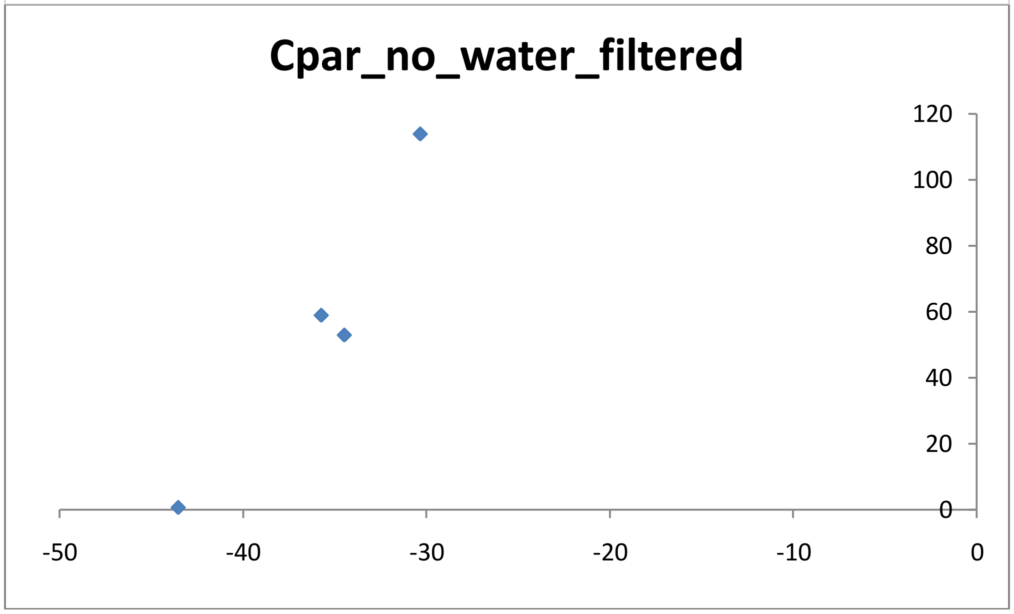


Trying to find a correlation from scores of wrong poses is impossible. As a result, if we filter out the compounds whose poses are clearly wrong, we might get a better correlation. Indeed, that is the case:



As a control, we also tested how the docking performance of a system that did not have a water in the rec.pdb file. These graphs are shown:

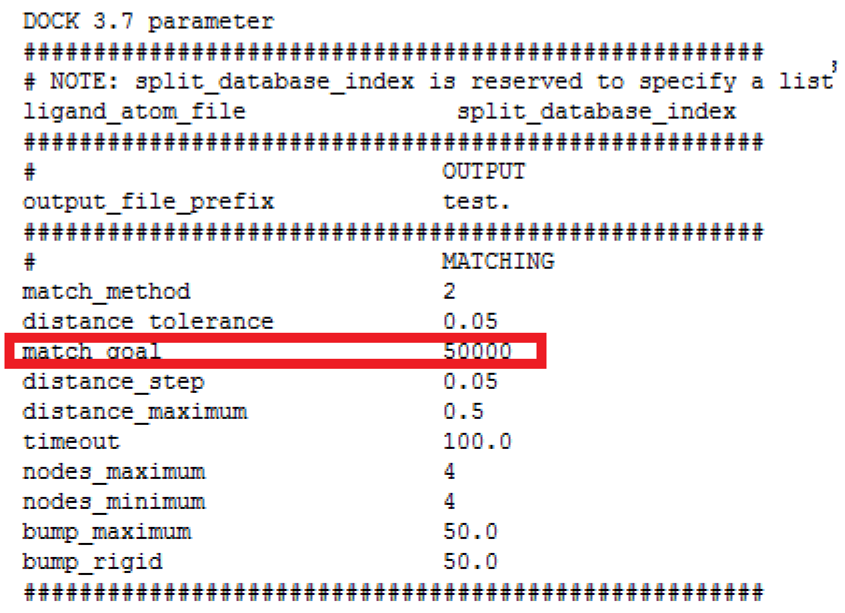




Clearly, docking performance improved, at least in terms of pose recapitulation, with the addition of the water. However, we shouldn't have to cut out all of our molecules.

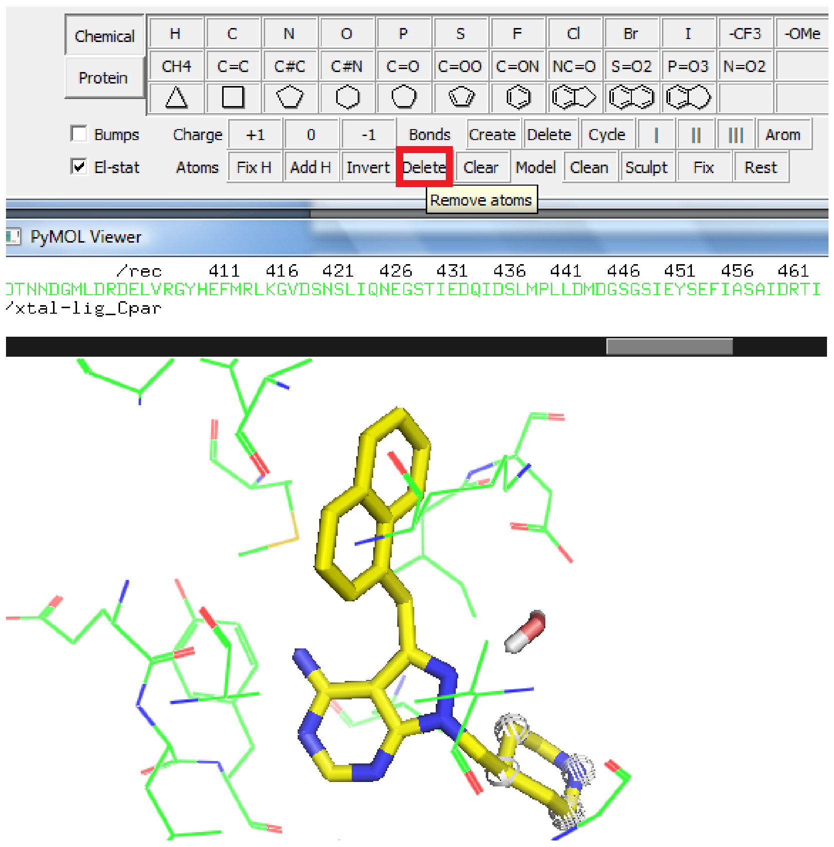
Two ways to improve docking performance are: 1) Increase sampling. This results in a slower run, but more possible poses will be scored. Perhaps this will increase the likelihood of finding the most correct pose. 2) Cut out the bottom adenine substituent from the xtal-lig.pdb file. Ligands that do not find correct poses are flipping out of the binding site in this region. It is possible that by not positioning the initial docking starting point here, more poses will explore the correct area.

To accomplish the increased sampling, the INDOCK file in the docking directory can be modified. Increasing the match_goal parameter increases the amount of acceptable conformations that are saved and scored before docking that molecule ends. Although match_goals of 500 or 5000 have been shown to be acceptable for virtual screening, let's try a match_goal of 50,000.

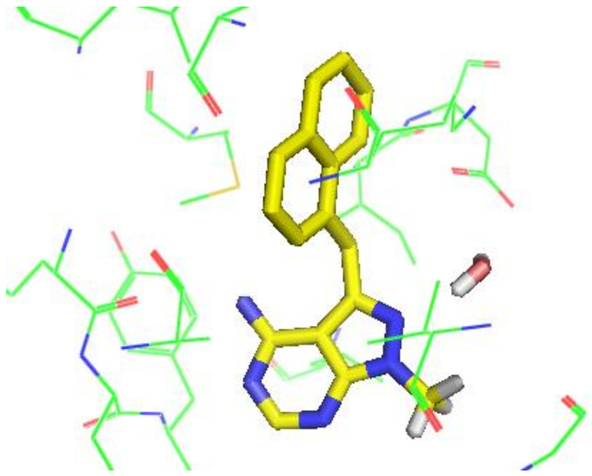


```
DOCK 3.7 parameter
#####################################################
# NOTE: split_database_index is reserved to specify a list
ligand_atom_file                split_database_index
#####################################################
#                              OUTPUT
output_file_prefix             test.
#####################################################
#                              MATCHING
match_method                   2
distance_tolerance             0.05
match_goal                     50000
distance_step                  0.05
distance_maximum               0.5
timeout                        100.0
nodes_maximum                  4
nodes_minimum                  4
bump_maximum                   50.0
bump_rigid                     50.0
#####################################################
```

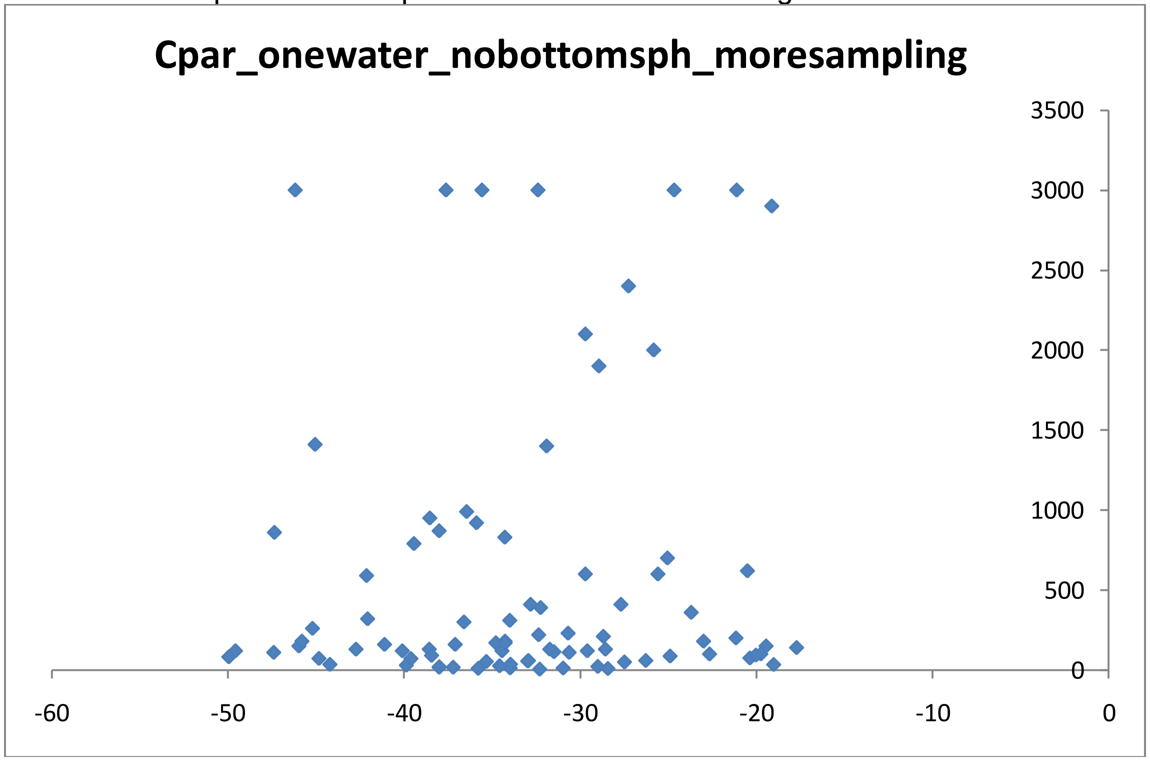
For the second consideration, we can open the *C. parvum* xtal-lig.pdb in PyMOL and edit it manually. Again, go into Builder, and click on the atoms of the ligand to delete.

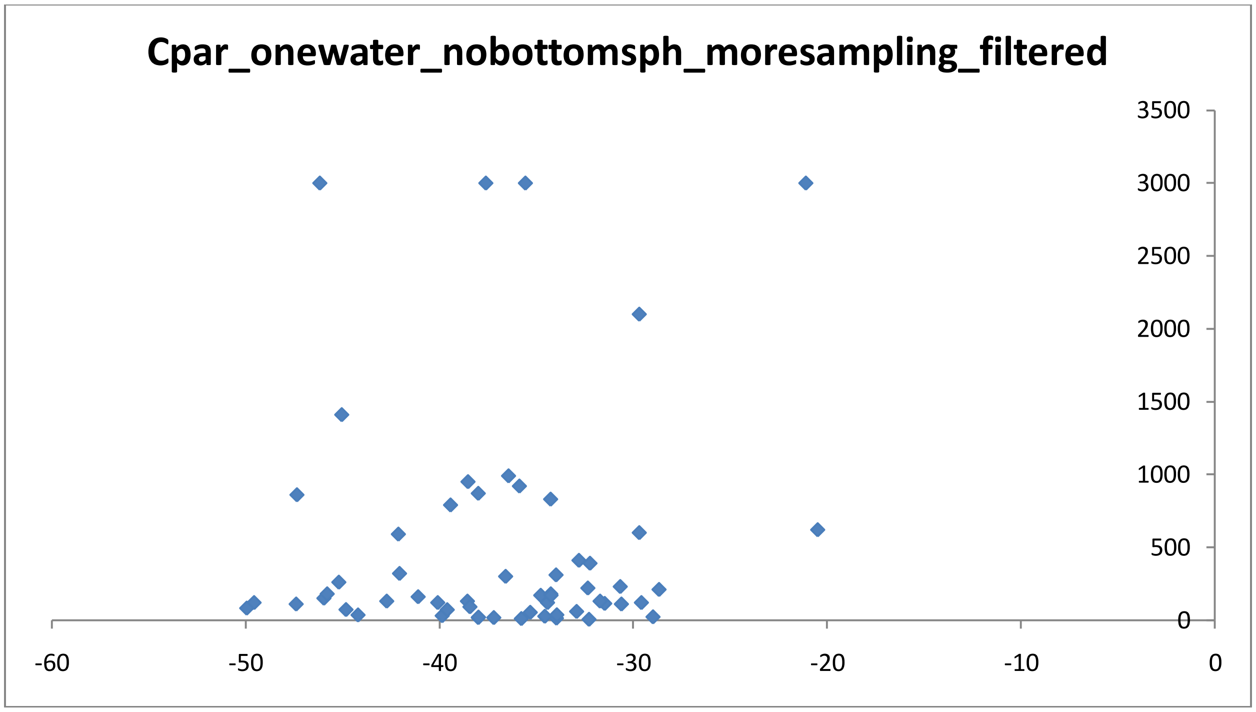


Then press the delete button:



Now, save this molecule as your new xtal-lig.pdb. Create a separate folder, (https://github.com/joelkarpiak/tdt_part3/Cpar_onewater_nobottomsph_moresampling), and re-run blastermaster.py with your new xtal-lig.pdb, old rec.pdb, and new INDOCK file. Once this is done, re-dock the lit compounds. The performance of this build is given here:





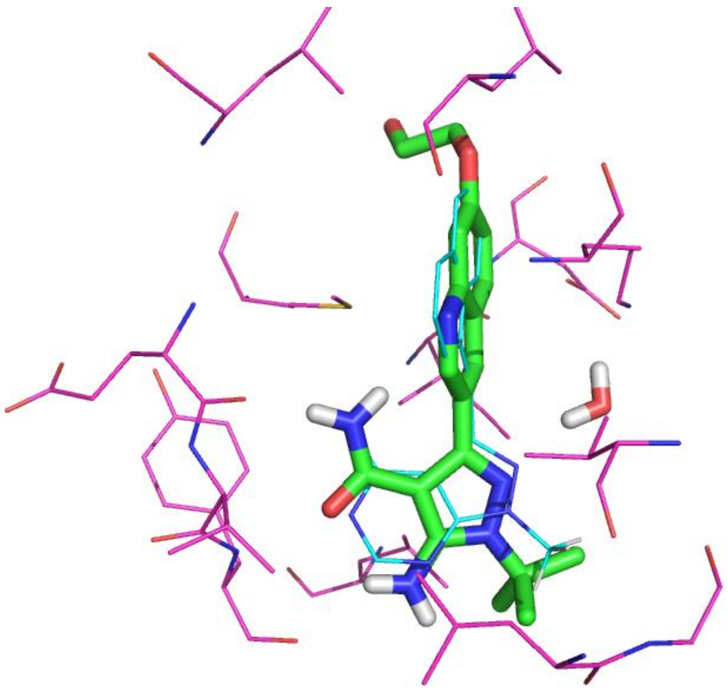
While predictive ability with IC_{50} did not increase, more compounds found correct poses. Also, since this is a merge of SAR with two completely different scaffolds, the data may seem more scattered than it actually is.

Since more compounds docked correctly with this set-up, let's keep it and apply the new xtal-lig.pdb and INDOCK to the *T. gondii* structure, since known compounds bind very similarly to both kinases. Blastermaster.py has to again be re-run, and for both the *C. parvum* and *T. gondii* new builds, we have to dock the 22 ligand test set using the same docking commands.

These results can be found at https://github.com/joelkarpiak/tdt_part3/Cpar_onewater_nobottomsph_moresampling and https://github.com/joelkarpiak/tdt_part3/Tgon_onewater_nobottomsph_moresampling. Each ligand set (lit and 22_ligs) is there with their poses (Cpar_22ligs_Cpar_onewater_nobottomsph_moresampling.mol2, Cpar_training_set_Cpar_onewater_nobottomsph_moresampling.mol2, Tgon_training_set_Cpar_onewater_nobottomsph_moresampling.mol2, and 22_ligs_Tgon.mol2), as well as the respective docking scores (extract_all.sort.uniq_Cpar_22ligs, extract_all.sort.uniq_lit_Cpar_training_set.txt, extract_all.sort.uniq_lit_Tgon_training_set.txt, extract_all.sort.uniq_Tgon_22ligs.mol2) for each ligand.

For *C. parvum*, the following molecules did not bind in seemingly correct poses: 1530, 1555, 1480, 1542, 1479, 1514, 1433, and 1563, so predictions about their relative affinities won't have the same confidence as others. Some, such as 1555, seem simply too big to adopt a seemingly correct pose, so they may be non-binders.

For *T. gondii,* the following molecules did not bind in seemingly correct poses: 1480, 1555, 1479, 1505, 1610, 1536. Also, even though the docking was based on the adenine-like compounds, DOCK 3.7 most likely correctly predicted the pose of compounds with different core structures, like 1570 shown here.

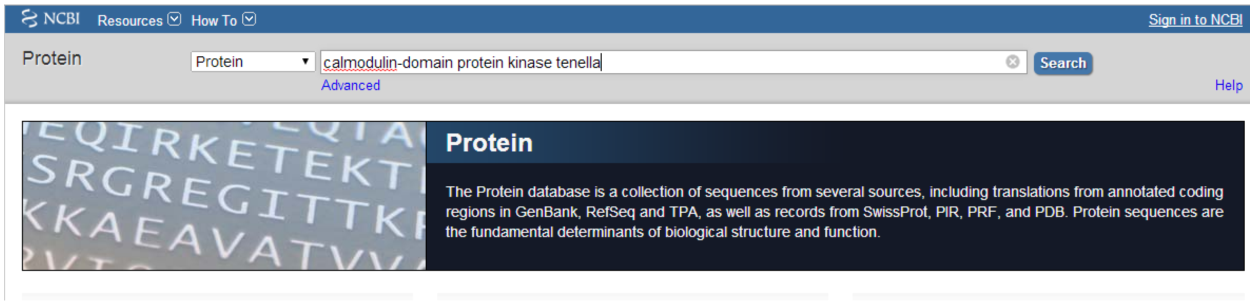


(b) Activity Prediction for CDPK1s with no known structure

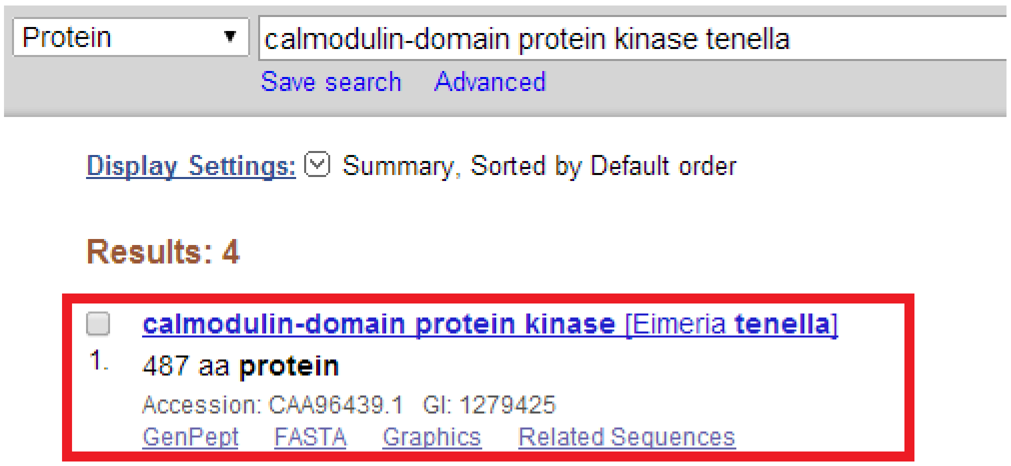
To predict the activity of these 22 ligands on *E. tenella,* first, a structure of this enzyme must be built, as there is no crystal structure. One way to do this is with the program MODELLER, and we'll be using the 9v8 version. MODELLER is free and easily installed and accessed from the Sali lab at UCSF:
https://salilab.org/modeller/download_installation.html

Once installed, the first thing needed is a sequence alignment between the proteins of interest and the most closely related protein with a structure to serve as a template. Here, we will use the *T. gondii* and *C. parvum* structures.

First, we need the sequences of the structures to be modeled. We can get these from the NCBI Protein resource, which has high-quality protein sequence information from many organisms (http://www.ncbi.nlm.nih.gov/protein/). Search for calmodulin-domain protein kinase tenella to find the sequence:



The Protein database is a collection of sequences from several sources, including translations from annotated coding regions in GenBank, RefSeq and TPA, as well as records from SwissProt, PIR, PRF, and PDB. Protein sequences are the fundamental determinants of biological structure and function.

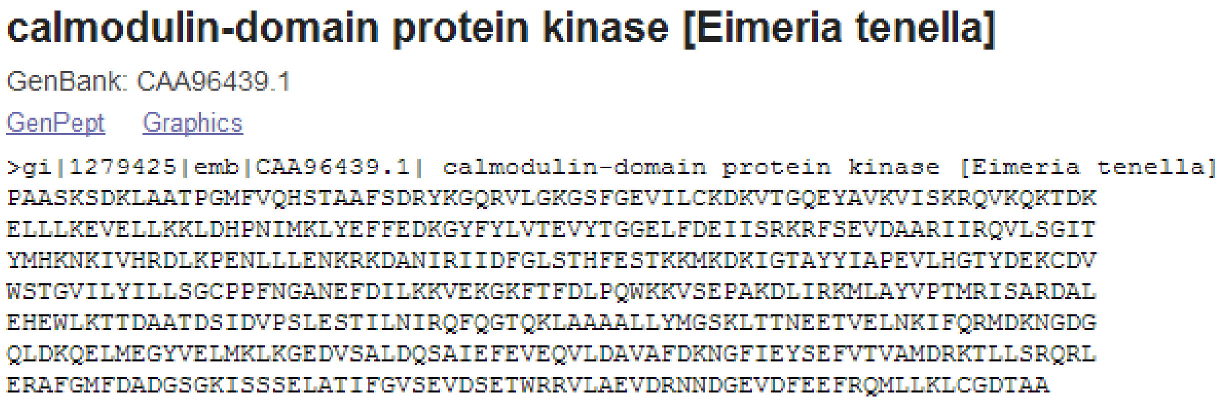


Summary, Sorted by Default order

Results: 4

1. calmodulin-domain protein kinase [Eimeria tenella]
487 aa protein
Accession: CAA96439.1 GI: 1279425
GenPept FASTA Graphics Related Sequences

And download the FASTA format sequence. This is what we will use for our alignment:



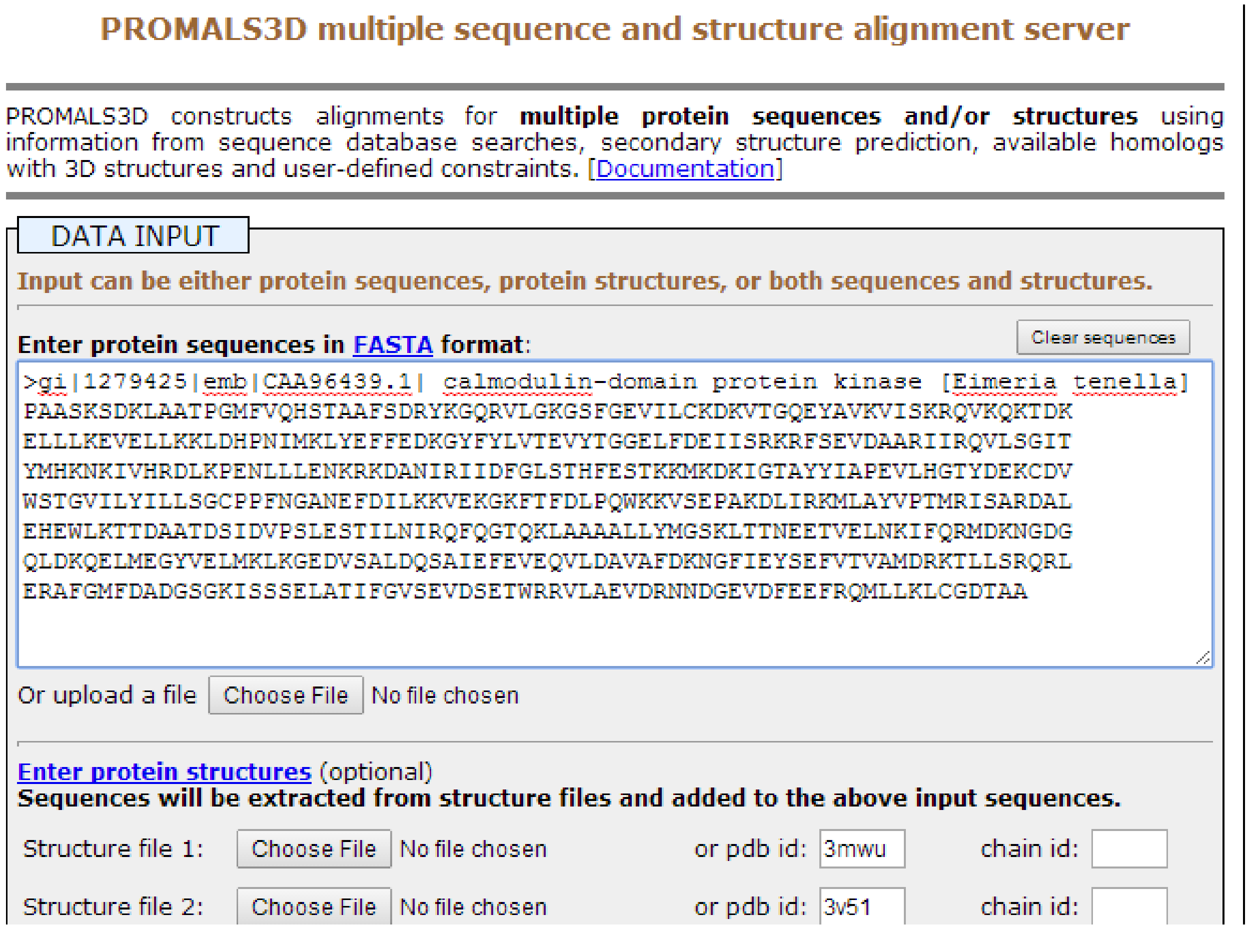
calmodulin-domain protein kinase [Eimeria tenella]

GenBank: CAA96439.1

GenPept Graphics

```
>gi|1279425|emb|CAA96439.1| calmodulin-domain protein kinase [Eimeria tenella]
PAASKSDKLAATPGMFVQHSTAAFSDRYKGQRVLGKGSFGEVILCKDKVTGQEYAVKVISKRQVKQKTDK
ELLLKEVELLKKLDHPNIMKLYEFFEDKGYFYLVTEVYTGGELFDEIISRKRFSEVDAARIIRQVLSGIT
YMHKNKIVHRDLKPENLLLENKRKDANIRIIDFGLSTHFESTKKMKDKIGTAYYIAPEVLHGTYDEKCDV
WSTGVILYILLSGCPPFNGANEFDILKKVEKGKFTFDLPQWKKVSEPAKDLIRKMLAYVPTMRISARDAL
EHEWLKTTDAATDSIDVPSLESTILNIRQFQGTQKLAAAALLYMGSKLTTNEETVELNKIFQRMDKNGDG
QLDKQELMEGYVELMKLKGEDVSALDQSAIEFEVEQVLDAVAFDKNGFIEYSEFVTVAMDRKTLLSRQRL
ERAFGMFDADGSGKISSSELATIFGVSEVDSETWRRVLAEVDRNNDGEVDFEEFRQMLLKLCGDTAA
```

To generate an alignment, we can use the free webserver PROMALS-3D, which takes into account the 3-D structure of the templates to produce a sequence alignment (http://prodata.swmed.edu/promals3d/promals3d.php). Input the sequences to model in FASTA form in the top box, then type in the PDB codes of the known templates below.


PROMALS3D multiple sequence and structure alignment server

PROMALS3D constructs alignments for **multiple protein sequences and/or structures** using information from sequence database searches, secondary structure prediction, available homologs with 3D structures and user-defined constraints. [Documentation]

DATA INPUT

Input can be either protein sequences, protein structures, or both sequences and structures.

Enter protein sequences in FASTA format:

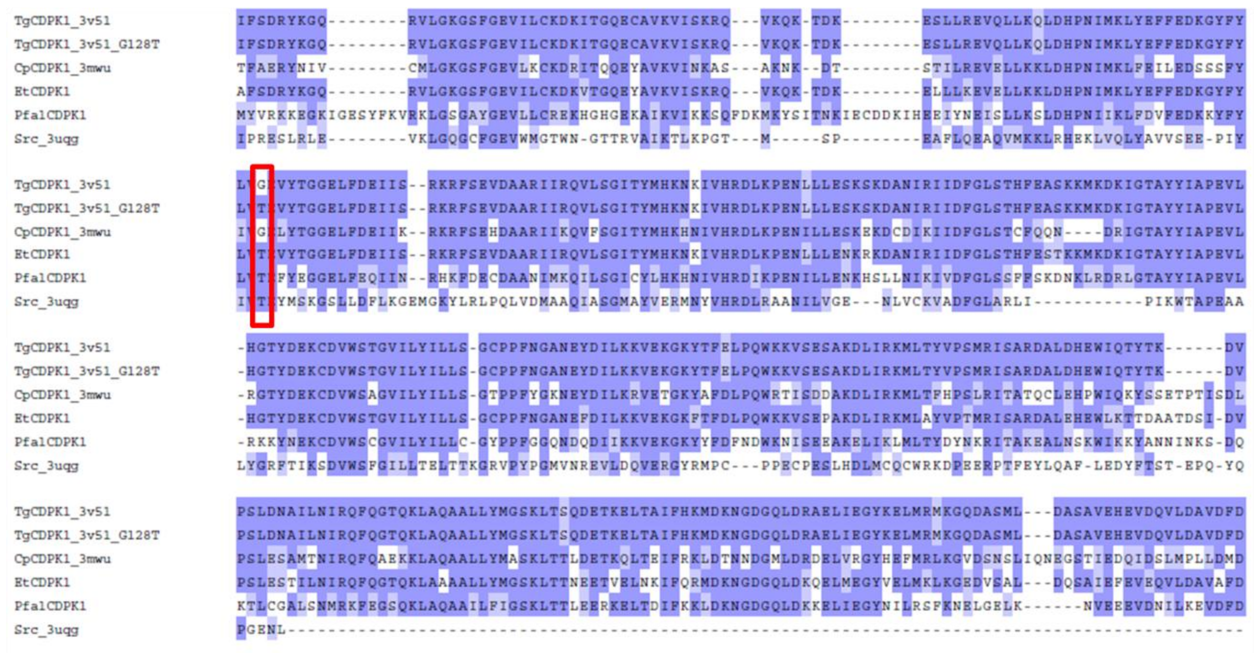
```
>gi|1279425|emb|CAA96439.1| calmodulin-domain protein kinase [Eimeria tenella]
PAASKSDKLAATPGMFVQHSTAAFSDRYKGQRVLGKGSFGEVILCKDKVTGQEYAVKVISKRQVKQKTDK
ELLLKEVELLKKLDHPNIMKLYEFFEDKGYFYLVTEVYTGGELFDEIISRKRFSEVDAARIIRQVLSGIT
YMHKNKIVHRDLKPENLLLENKRKDANIRIIDFGLSTHFESTKKMKDKIGTAYYIAPEVLHGTYDEKCDV
WSTGVILYILLSGCPPFNGANEFDILKKVEKGKFTFDLPQWKKVSEPAKDLIRKMLAYVPTMRISARDAL
EHEWLKTTDAATDSIDVPSLESTILNIRQFQGTQKLAAAALLYMGSKLTTNEETVELNKIFQRMDKNGDG
QLDKQELMEGYVELMKLKGEDVSALDQSAIEFEVEQVLDAVAFDKNGFIEYSEFVTVAMDRKTLLSRQRL
ERAFGMFDADGSGKISSSELATIFGVSEVDSETWRRVLAEVDRNNDGEVDFEEFRQMLLKLCGDTAA
```

Or upload a file No file chosen

Enter protein structures (optional)
Sequences will be extracted from structure files and added to the above input sequences.

Structure file 1:	Choose File	No file chosen	or pdb id:	3mwu	chain id:	
Structure file 2:	Choose File	No file chosen	or pdb id:	3v51	chain id:	

After submitting the job, a sequence alignment will eventually be produced (https://github.com/joelkarpiak/tdt_part3/ali1.fa). This alignment can be viewed using the free tool PFAAT (http://pfaat.sourceforge.net/). Simply launch the app, and then open the PROMALS3D-produced alignment in FASTA form, with the gatekeeper residue boxed in red. Also included is *P. falciparum* CDPK1 and human src, kinases with a threonine gatekeeper:



```
TgCDPK1_3v51         IFSDRYKGQ--------RVLGKGSFGEVILCKDKITGQECAVKVISKRQ---VKQK-TDK--------ESLLREVQLLKQLDHPNIMKLYEFFEDKGYFY
TgCDPK1_3v51_G128T   IFSDRYKGQ--------RVLGKGSFGEVILCKDKITGQECAVKVISKRQ---VKQK-TDK--------ESLLREVQLLKQLDHPNIMKLYEFFEDKGYFY
CpCDPK1_3mwu         TFAERYNIV--------CMLGKGSFGEVLKCKDRITQQEYAVKVINKAS---AKNK--DT--------STILREVELLKKLDHPNIMKLFEILEDSSSFY
EtCDPK1              AFSDRYKGQ--------RVLGKGSFGEVILCKDKVTGQEYAVKVISKRQ---VKQK-TDK--------ELLLKEVELLKKLDHPNIMKLYEFFEDKGYFY
PfalCDPK1            MYVRKKEGKIGESYFKVRKLGSGAYGEVLLCREKHGHGEKAIKVIKKSQFDKMKYSITNKIECDDKIHEEIYNEISLLKSLDHPNIIKLFDVFEDKKYFY
Src_3uqg             IPRESLRLE--------VKLGQGCFGEVWMGTWN-GTTRVAIKTLKPGT---M-----SP--------EAFLQEAQVMKKLRHEKLVQLYAVVSEE-PIY

TgCDPK1_3v51         LVGEVYTGGELFDEIIS--RKRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLESKSKDANIRIIDFGLSTHFEASKKMKDKIGTAYYIAPEVL
TgCDPK1_3v51_G128T   LVTEVYTGGELFDEIIS--RKRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLESKSKDANIRIIDFGLSTHFEASKKMKDKIGTAYYIAPEVL
CpCDPK1_3mwu         IVGELYTGGELFDEIIK--RKRFSEHDAARIIKQVFSGITYMHKHNIVHRDLKPENILLESKEKDCDIKIIDFGLSTCFQQN----DRIGTAYYIAPEVL
EtCDPK1              LVTEVYTGGELFDEIIS--RKRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLENKRKDANIRIIDFGLSTHFESTKKMKDKIGTAYYIAPEVL
PfalCDPK1            LVTEFYEGGELFEQIIN--RHKFDECDAANIMKQILSGICYLHKHNIVHRDIKPENILLENKHSLLNIKIVDFGLSSFFSKDNKLRDRLGTAYYIAPEVL
Src_3uqg             IVTEYMSKGSLLDFLKGEMGKYLRLPQLVDMAAQIASGMAYVERMNYVHRDLRAANILVGE---NLVCKVADFGLARLI-----------PIKWTAPEAA

TgCDPK1_3v51         -HGTYDEKCDVWSTGVILYILLS-GCPPFNGANEYDILKKVEKGKYTFELPQWKKVSESAKDLIRKMLTYVPSMRISARDALDHEWIQTYTK------DV
TgCDPK1_3v51_G128T   -HGTYDEKCDVWSTGVILYILLS-GCPPFNGANEYDILKKVEKGKYTFELPQWKKVSESAKDLIRKMLTYVPSMRISARDALDHEWIQTYTK------DV
CpCDPK1_3mwu         -RGTYDEKCDVWSAGVILYILLS-GTPPFYGKNEYDILKRVETGKYAFDLPQWRTISDDAKDLIRKMLTFHPSLRITATQCLEHPWIQKYSSETPTISDL
EtCDPK1              -HGTYDEKCDVWSTGVILYILLS-GCPPFNGANEFDILKKVEKGKFTFDLPQWKKVSEPAKDLIRKMLAYVPTMRISARDALEHEWLKTTDAATDSI-DV
PfalCDPK1            -RKKYNEKCDVWSCGVILYILLC-GYPPFGGQNDQDIIKKVEKGKYYFDFNDWKNISEEAKELIKLMLTYDYNKRITAKEALNSKWIKKYANNINKS-DQ
Src_3uqg             LYGRFTIKSDVWSFGILLTELTTKGRVPYPGMVNREVLDQVERGYRMPC---PPECPESLHDLMCQCWRKDPEERPTFEYLQAF-LEDYFTST-EPQ-YQ

TgCDPK1_3v51         PSLDNAILNIRQFQGTQKLAQAALLYMGSKLTSQDETKELTAIFHKMDKNGDGQLDRAELIEGYKELMRMKGQDASML---DASAVEHEVDQVLDAVDFD
TgCDPK1_3v51_G128T   PSLDNAILNIRQFQGTQKLAQAALLYMGSKLTSQDETKELTAIFHKMDKNGDGQLDRAELIEGYKELMRMKGQDASML---DASAVEHEVDQVLDAVDFD
CpCDPK1_3mwu         PSLESAMTNIRQFQAEKKLAQAALLYMASKLTTLDETKQLTEIFRKLDTNNDGMLDRDELVRGYHEFMRLKGVDSNSLIQNEGSTIEDQIDSLMPLLDMD
EtCDPK1              PSLESTILNIRQFQGTQKLAAAALLYMGSKLTTNEETVELNKIFQRMDKNGDGQLDKQELMEGYVELMKLKGEDVSAL---DQSAIEFEVEQVLDAVAFD
PfalCDPK1            KTLCGALSNMRKFEGSQKLAQAAILFIGSKLTTLEERKELTDIFKKLDKNGDGQLDKKELIEGYNILRSFKNELGELK------NVEEEVDNILKEVDFD
Src_3uqg             PGENL----------------------------------------------------------------------------------------------
```

E. tenella is very close at 86% sequence identity, which means that these proteins are certainly able to be modeled with confidence.

To do so, delete every sequence besides the template and the desired protein. For *E. tenella*, you'll get a file like (alignment files all at https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking):

>3v51_chainA_p010
...................STAIFSDRYKGQRVLGKGSFGEVILCKDKITGQECAVKVIS
KRQVKQKTDKESLLREVQLLKQLDHPNIMKLYEFFEDKGYFYLVGEVYTGGELFDEIISR
KRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLESKSKDANIRIIDFGLSTHFE
ASKKMKDKIGTAYYIAPEVLHGTYDEKCDVWSTGVILYILLSGCPPFNGANEYDILKKVE
KGKYTFELPQWKKVSESAKDLIRKMLTYVPSMRISARDALDHEWIQTYTK.....DVPSL
DNAILNIRQFQGTQKLAQAALLYMGSKLTSQDETKELTAIFHKMDKNGDGQLDRAELIEG
YKELMRMKGQDASMLDASAVEHEVDQVLDAVDFDKNGYIEYSEFVTVAMDRKTLLSRERL
ERAFRMFDSDNSGKISSTELATIFGVSDVDSETWKSVLSEVDK.NDGEVDFDEFQQMLLK
LCGN...
>gi_1279425_emb_CAA96439.1
PAASKSDKLAATPGMFVQHSTAAFSDRYKGQRVLGKGSFGEVILCKDKVTGQEYAVKVIS
KRQVKQKTDKELLLKEVELLKKLDHPNIMKLYEFFEDKGYFYLVTEVYTGGELFDEIISR
KRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLENKRKDANIRIIDFGLSTHFE
STKKMKDKIGTAYYIAPEVLHGTYDEKCDVWSTGVILYILLSGCPPFNGANEFDILKKVE
KGKFTFDLPQWKKVSEPAKDLIRKMLAYVPTMRISARDALEHEWLKTTDAATDSIDVPSL
ESTILNIRQFQGTQKLAAAALLYMGSKLTTNEETVELNKIFQRMDKNGDGQLDKQELMEG
YVELMKLKGEDVSALDQSAIEFEVEQVLDAVAFDKNGFIEYSEFVTVAMDRKTLLSRQRL
ERAFGMFDADGSGKISSSELATIFGVSEVDSETWRRVLAEVDRNNDGEVDFEEFRQMLLK
LCGDTAA

MODELLER requires a specific format (ali) for alignments that looks like (file named, ali1.ali):

```
>P1;3v51
structureX:3v51.pdb:  45:A:+507 : :::-1.00:-1.00
AIFSDRYKGQRVLGKGSFGEVILCKDKITGQECAVKVIS
KRQVKQKTDKESLLREVQLLKQLDHPNIMKLYEFFEDKGYFYLVGEVYTGGELFDEIISR
KRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLESKSKDANIRIIDFGLSTHFE
ASKKMKDKIGTAYYIAPEVLHGTYDEKCDVWSTGVILYILLSGCPPFNGANEYDILKKVE
KGKYTFELPQWKKVSESAKDLIRKMLTYVPSMRISARDALDHEWIQTYTKEQISVDVPSL
DNAILNIRQFQGTQKLAQAALLYMGSKLTSQDETKELTAIFHKMDKNGDGQLDRAELIEG
YKELMEHEVDQVLDAVDFDKNGYIEYSEFVTVAMDRKTLLSRERL
ERAFRMFDSDNSGKISSTELATIFGVSDVDSETWKSVLSEVDKNNDGEVDFDEFQQMLLK
LCGN---*
>P1;Et_CDPK1
sequence::     : :     : :::-1.00:-1.00
AAFSDRYKGQRVLGKGSFGEVILCKDKVTGQEYAVKVIS
KRQVKQKTDKELLLKEVELLKKLDHPNIMKLYEFFEDKGYFYLVTEVYTGGELFDEIISR
KRFSEVDAARIIRQVLSGITYMHKNKIVHRDLKPENLLLENKRKDANIRIIDFGLSTHFE
STKKMKDKIGTAYYIAPEVLHGTYDEKCDVWSTGVILYILLSGCPPFNGANEFDILKKVE
KGKFTFDLPQWKKVSEPAKDLIRKMLAYVPTMRISARDALEHEWLKTTDAATDSIDVPSL
ESTILNIRQFQGTQKLAAAALLYMGSKLTTNEETVELNKIFQRMDKNGDGQLDKQELMEG
YVELMEFEVEQVLDAVAFDKNGFIEYSEFVTVAMDRKTLLSRQRL
ERAFGMFDADGSGKISSSELATIFGVSEVDSETWRRVLAEVDRNNDGEVDFEEFRQMLLK
LCGDTAA*
```

Here, gaps are replaced by dashes, an asterisk * denotes the end of the sequence, and the template information, at the top, shows which structure file to use as input for modeling, as well as the start and stop residue numbers. Here, the plain 3v51.pdb file will be used. To run MODELLER, all we need now is a python file with instructions on what to build. This file looks like this (model.py):

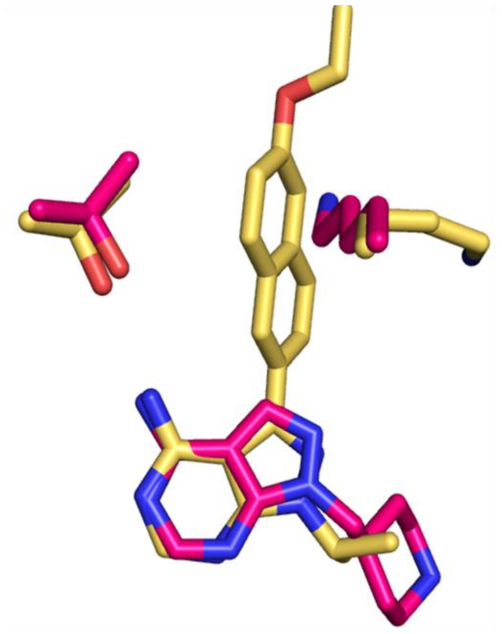
```python
from modeller import *
from modeller.automodel import *

env = environ()
a = automodel(env, alnfile='ali1.ali',
            knowns='3v51', sequence='Et_CDPK1',
            assess_methods=(assess.DOPE))
a.starting_model = 1
a.ending_model = 5
a.make()
```

Here, we are making 5 models of Et_CDPK1, using an input alignment file ali1.ali and template 3v51. Running the program via the command:

```
'/path/to/modeler/mod9v8 model.py'
```

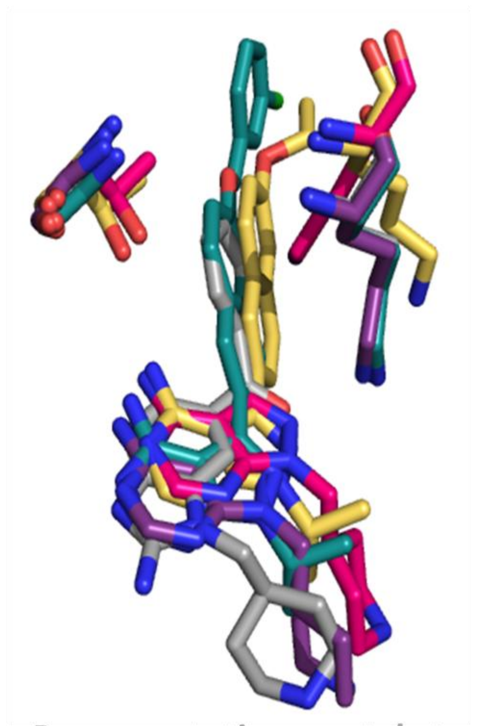
will build 5 models of *E. tenella* CDPK1.

To evaluate these models, we need to compare them against known crystal structures. Although *E. tenella* is extremely close in sequence to *T. gondii*, one key difference is the threonine gatekeeper. So, we can look at crystal structures of human Src, in complex with BKIs and containing a threonine gatekeeper, to pick the correct threonine rotamer. Comparison of Src crystal structures 3uqg 3uqf show a consistent threonine rotamer, which is present in the *E. tenella* models:



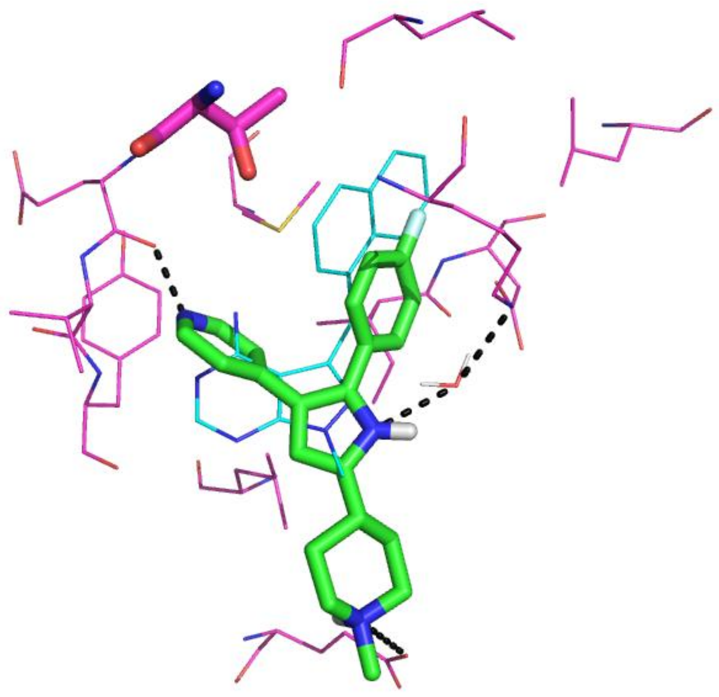
Crystal structures of human Src

However, the lysine shifts away if there is no bulky group occupying the space near the gatekeeper. To see if this is a lysine rotamer that is consistently present, we looked at *T. gondii* and *C. parvum* crystal structures and compared them to Src; none of them featured this flipped-inward lys:


Representative crystal structures

Of the five models that were generated, only one model had these chosen lys and thr rotamers: model 2 (https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/Et_CDPK1.B99990002_winner.pdb).

For docking to *E. tenella*, we will be using the same water, xtal-lig.pdb, and sampling parameters as in the previous *T. gondii* and *C. parvum* prospective studies. Again, set-up the docking by using blastermaster.py, and then dock, as previously, the 22-ligand test set. Here, compound numbers 1610, 1479, 1480, and 1478 did not dock correctly. (https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/22ligs_Et.mol2 and https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/ extract_all.sort.uniq_22_ligs.txt are the relative scores). Also, two compounds are known to bind to *E. tenella*[1] CDPK1, which were the original promiscuous kinase inhibitors discovered. These 2 were docked using the same protocol, and the most likely binding mode, which looks reasonable, is shown here:



Here, even though the placement of the conserved water was a guess, it seems to have been a good one. It is easily coordinating the polar group of the ligand and the lys in the correct rotamer. The bottom charged substituent ring is interacting with the conserved glu residue, and another ring overlaps with the normal adenine ring to interact with the hinge region. Due to the bulky gatekeeper residue, though, the normal plane of the substituted phenyl ring has been repulsed and pushed back closer to the conserved lys (https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/Et_ligs.mol2).

2. Follow-up hit finding

a) Building eMolecules small molecule docking databases

The last step is to dock the large e-molecules databse. Docking this large database is just as easy as docking the small databases, 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. We will however, only use 5000 as the match goal instead of the 50000 used to generate the best poses for the test set, purely in the interest of computer time.

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 *E. tenella*

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

`setup_db2.py /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).

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. Most of the polar groups of the docked molecules are engaged with complementary polar groups in the protein. Additionally, the putative ligands fill 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. 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 the SAMPL4 challenge 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.

Both raw output files from our runs are here:
https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/extract_all.sort.uniq_frags_emolecules.txt with respective poses at
https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/Et_emolcules_frags_top1000,mol2;
https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/extract_all.sort.uniq_emolecules_leads.txt, with respective poses at
https://github.com/joelkarpiak/tdt_part3/EtCDPK1_docking/Et_emolcules_leads_top1000,mol2).

Adding this tutorial https://github.com/joelkarpiak/tdt_part3/tutorial.pdf, once completed, to the submission files, results in the submitted tarball:
https://github.com/joelkarpiak/tdt_part3/submission.tar.gz

Specific files to include for this challenge

1. Predictions for held-out, external test set against CDPK1 in T. gondii, C. parvum and E. tenella (identifier, smiles, and measure of predicted activity to allow rank-order):
https://github.com/joelkarpiak/tdt_part3/Tgon_22.id
https://github.com/joelkarpiak/tdt_part3/Cparv.id
https://github.com/joelkarpiak/tdt_part3/Eten.id
2. Rank-ordered list of top-1000 commercial compounds predicted to be active against E. tenella CDPK1(identifier, smiles)
https://github.com/joelkarpiak/tdt_part3/Eten_leads.id
https://github.com/joelkarpiak/tdt_part3/Eten_frags.id

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

[1]Donald, et al. Anticoccidal kinase inhibitors: Identification of protein kinase targets secondary to cGMP-dependent protein kinase. 2006. http://www.sciencedirect.com/science/article/pii/S0166685106001447

Murphy, et al. Discovery of potent and selective inhibitors of CDPK1 from C. parvum and T. gondii. 2010. http://pubs.acs.org/doi/abs/10.1021/ml100096t

Zhang, et al. Benzoylbenzimidazole-based selective inhibitors targeting Cryptosporidium parvum and Toxoplasma gondii calcium-dependent protein kinase-1. 2012. http://www.ncbi.nlm.nih.gov/pubmed/22795629