Custom Scoring CDPK1 with SMINA

David Ryan Koes
dkoes@pitt.edu
Department of Computational and Systems Biology
University of Pittsburgh

In this tutorial we will describe how to use SMINA to perform structure-based virtual screening against the calcium-dependent protein kinase 1 (CDPK1) of *E. tenella*. *E. tenella* is a parasite that infects young poultry and results in potentially fatal coccidiosis. Although no structure is available for this enzyme, structures of the enzyme in related parasites, *C. parvum* and *T. gondii*, are available, and there is bioactivity data available for small molecules active against these enzymes.

We will explain how to analyze these known structures, evaluate the performance of docking with SMINA against these structures, create a custom scoring function using these docking results, create a structural model of *E. tenella* with SWISS-MODEL, and perform a virtual screen against *E. tenella* CDPK1.

Assumptions and Conventions We assume that the reader is familiar with the Linux command-line and is able to install the necessary software. The tutorial was implemented and tested on Ubuntu Linux 12.04. Commands for the bash command-line environment are displayed:

```
echo "Hello"
```

Commands for the R statistical computing environment are displayed:

```
print('Hello')
```

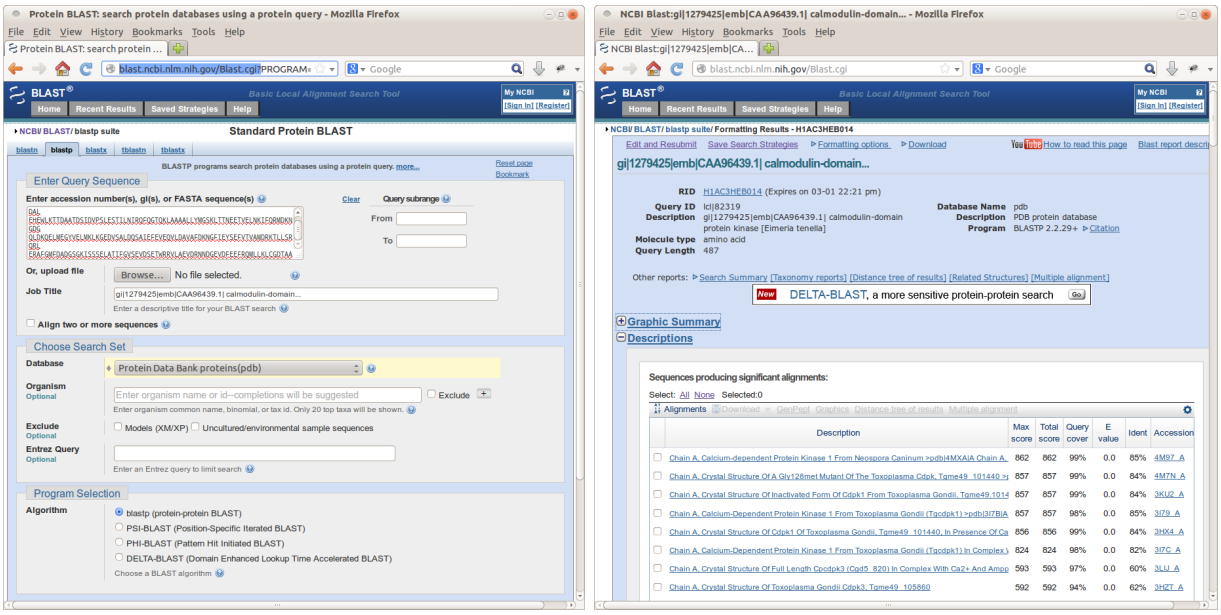
Commands for the python-based PyMOL environment are displayed:

```
print "Hello"
```

Software	License	Source
color_by_mutation.py	GNU	http://www.pymolwiki.org/index.php/Color_By_Mutations
PyMOL v1.6.0	Python	http://sourceforge.net/projects/pymol/
R v3.0.3	GPLv2	http://r-project.org
RDKit 2012_06_1	BSD Style	http://rdkit.org/
sdsorter 2014-02-19	GPLv2	http://sourceforge.net/projects/sdsorter/
smina [ab615b]	GPLv2	http://sourceforge.net/projects/smina/

1 Validation of *C. parvum* and *T. gondii* crystal structures

First we analyze the *E. tenella* sequence and examine the structural differences and similarities between the three targets. We then create a virtual screening benchmark using available binding data for *C. parvum* and *T. gondii* and dock these benchmark compounds. We use the results of docking to select crystal structures for virtual screening and additional modeling.



(a) (b)

Figure 1: BLASTing *E. tenella*. (a) In the search screen we provide the *E. tenella* sequence and specify the Protein Data Bank database. (b) The results show the available structures with high sequence identity to *E. tenella*.

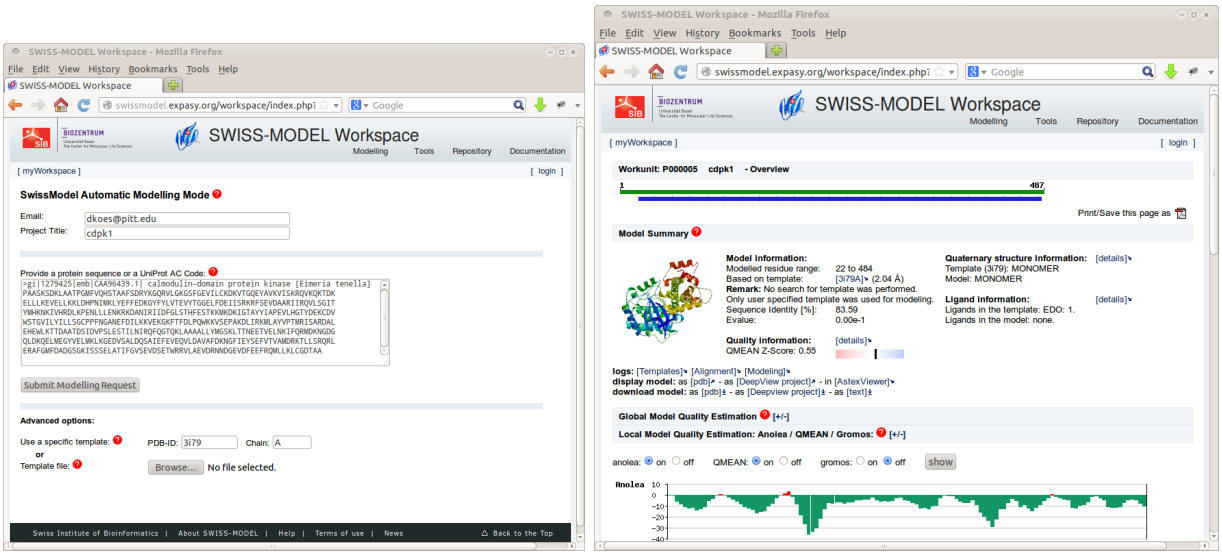
1.1 Target Analysis

First, we analyze the provided *E. tenella* sequence and *C. parvum* and *T. gondii* structures.

1.1.1 *E. tenella* Sequence Analysis

1. Copy or download the FASTA sequence for *E. tenella* (http://www.ncbi.nlm.nih.gov/protein/CAA96439).
2. Start a protein BLAST search http://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastp.
3. Paste or upload the FASTA sequence and change the database to Protein Data Bank (see Figure 1(a)).
4. Run the BLAST search.
5. Analyze the results (Figure 1(b)). The top ranked hit is from *Neospora Caninum* (4M97), but it is closely followed by several structures from *T. gondii* (4M7N, 3KU2, 3I79, 3HX4, 3I7C) with >80% sequence identity. This first *C. parvum* hit (3IGO) has 62% sequence identity.

From this analysis we conclude that CDPK1 of *T. gondii* has a higher sequence identity to *E. tenella* compared to *C. parvum*. This suggests that a *T. gondii* structure is the more logical choice for homology modeling.



(a) (b)

Figure 2: Homology modelling *E. tenella*. (a) In the search screen we provide the *E. tenella* sequence and specify 3I79 from *T. gondii* as our template structure. (b) The results show the available structures with high sequence identity to *E. tenella*.

1.1.2 Structure Analysis

In order to identify the CDPK1 binding site differences between *T. gondii* and *E. tenella*, we create a homology model for *E. tenella* and use `color_by_mutation.py` and PyMol to quickly visualize the sequence differences on the structures.

1. Go to SWISS-MODEL [2] (`http://swissmodel.expasy.org/`) and click on Automated Mode.
2. Input the FASTA for *E. tenella* and specify 3I79 as a template (Figure 2(a)).
3. Download the PDB file from the results (Figure 2(b)).
4. Load the model, 3I79, and 3NCG into PyMOL.

```
load swissmodel.pdb
fetch 3I79
fetch 3NCG
```

5. Align the structures.

```
alignto
```

6. Extract the ligand BK1 from 3NCG and hide the rest of the structure. Since 3I79 is unbound, we will use this ligand to delineate the binding site.

```
extract bk1,resn BK1
hide (/3NCG/)
```

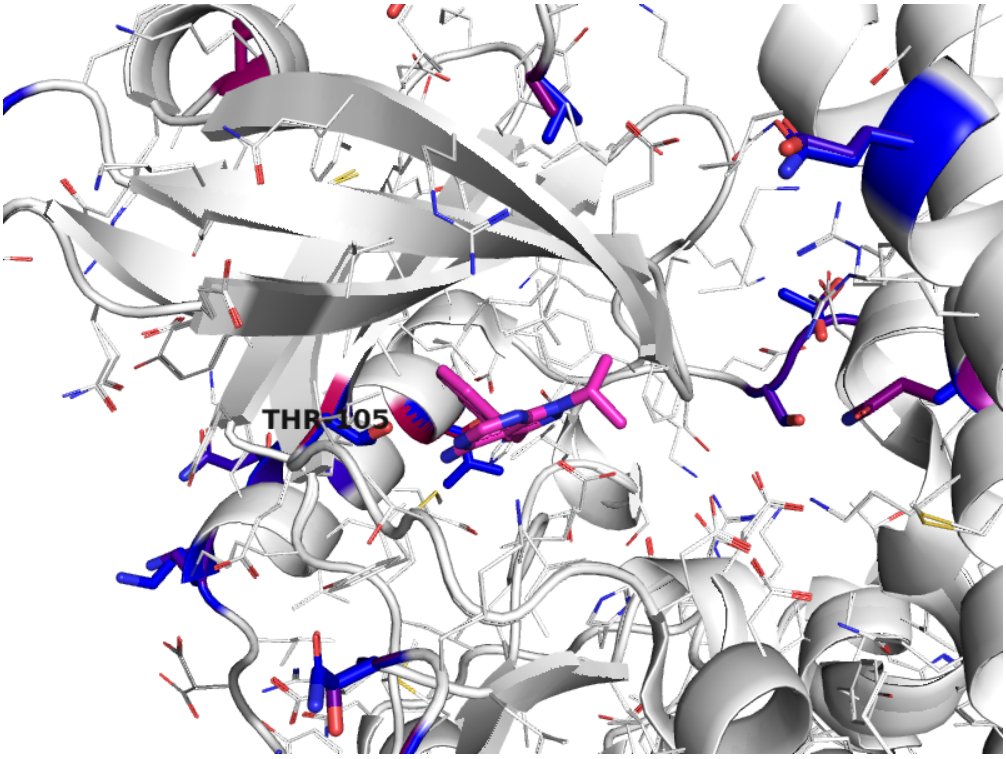


Figure 3: The binding site of the *E. tenella* model compared to the 3I79 structure of *T. gondii*.

7. Inspect the residues of the model and 3I79 around the binding site. Notice that SWISS-MODEL has faithfully duplicated the side chain conformations.

8. Color the model and 3I79 by mutation.

```
run color_by_mutation.py
color_by_mutation.py swissmodel,3i79
```

9. Inspect the binding site. Notice that GLY128 in 3I79 is mutated to THR 105 in the *E. tenella* model (see Figure 3). The rest of the binding site is largely unchanged.

10. Repeat this analysis for 3NCG. Notice that this *C. parvum* structure has several more mutations in the binding site.

1.2 Benchmark Creation

Assays against both *T. gondii* and *C. parvum* CDPK1 are published in PubChem. Here we will use this data to construct virtual screening benchmarks against these two targets.

1. **Assemble active compounds**

   (a) Identify actives in PubChem (http://pubchem.ncbi.nlm.nih.gov). Search assays for CDPK1 and sort the results by number of actives. Choose the three largest biochemical

assays for each target.
T. gondii:
http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=677062
http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=600378
http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=672956
C. parvum:
http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=600379
http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=672955
http://pubchem.ncbi.nlm.nih.gov/assay/assay.cgi?aid=66097

(b) Download actives. Click on the SDF download button and choose 'Structures (SDF,etc.) - Active'. Select uncompressed SMILES format.

(c) Fix SMILES. The downloaded file has the SMILES string and title in the wrong order. Additionally, we need to remove salts and append the keyword 'active' to the title of all active compounds. For each downloaded file, e.g. `aid677062.txt`, execute:

```
awk '{print $2,$1"_active"}' aid677062.txt | \
        sed 's/\.Cl//' > aid677062.smi
```

(d) Combine actives into a single file for each target.

```
cat aid*.smi > actives.smi
```

There should 156 actives for *T. gondii* and 89 actives for *C. parvum*.

2. **Create Decoy Set** Since there is very little data on inactive compounds for these targets, we will generate a set of inactive decoys using the Database of Useful Decoys: Enhanced (DUDE) [4] method for sampling decoys from the ZINC database [1]. These decoys are selected to be chemically dissimilar from the provided active compounds (and therefore unlikely to bind) while still matching simple molecular properties such as molecular weight, calculated logP, and the number of rotatable bonds and hydrogen bond acceptors/donors.

   (a) Create decoy set from active set using the DUDE website (http://dude.docking.org/generate). This will produce a tarball, `dude-decoys.tar.gz`, that contains 50 property-matched decoys for each provided active in the subdirectory `decoys`.

   (b) Combine decoys into a single file.

   ```
   cat decoys/* | grep -v ligand > decoys.smi
   ```

3. **Generate Conformers** We will used the open-source RDKit library to generate 3D conformations from the 2D SMILES.

   (a) Having installed RDKit and placed it in your `PYTHONPATH`, download and make executable the `rdconf.py` python script.

   ```
   wget http://bits.csb.pitt.edu/tdtCDPK1/rdconf.py
   chmod +x rdconf.py
   ```

   (b) Generate a single conformer for each active/decoy.

```
rdconf.py --maxconfs 1 decoys.smi decoys.sdf
rdconf.py --maxconfs 1 actives.smi actives.sdf
```

(c) Create a single combined file of both actives and decoys.

```
cat actives.sdf decoys.sdf > combined.sdf
```

1.3 Docking

We will perform virtual screening by docking compounds to a receptor structure using SMINA [3] and the AutoDock Vina scoring function [6]. Since we a docking against a fixed receptor, it is important that we choose a good receptor structure. Here we will use our CDPK1 benchmark to simultaneously evaluate our docking protocol and choice of receptor structure.

1. Identify all *C. parvum* and *T. gondii* structures in the PDB. Search for 'calcium-dependent protein kinase 1' and then narrow the search by the appropriate organism.
   C. parvum: *2QG5* 2WEI *3DFA* 3F3Z 3HKO 3IGO *3L19* 3LIJ 3MWU 3NCG
   T. gondii: 3I79 3I7B 3I7C 3KU2 3N51 3NYV 3SX9 3SXF 3T3U 3T3V 3UPX 3UPZ 3V51 3V5P 3V5T 4M84

2. Download these structures to pdb files.

3. Align and extract structures. Open all the pdb files of a target in pymol. Align them (`alignto`). Remove waters and ions. Extract each ligand into its own object.

4. Save aligned receptor and ligand files. Note that 3LI9 is missing the kinase domain and 2DFA and 2QG5 are unbound structures of *C. parvum* so we will omit them from further consideration.

   ```
   for name in cmd.get_names(): cmd.save(name+".pdb",name)
   ```

5. Combine extracted ligands into a single pdb to form a pseudo-molecule that defines the binding site.

   ```
   grep -h HETATM obj*pdb > allligs.pdb
   ```

6. Dock the benchmark compounds to each receptor structure.[1] For example:

   ```
   smina --seed 0 --autobox_ligand allligs.pdb -r 2WEI.pdb \
         -l combined.sdf -o 2WEI_docked.sdf.gz
   ```

   For reproducibility, we specify a random number seed. The bounding box for docking is specified automatically with the `autobox_ligand` option which creates a box with an 8Å buffer around the provided ligand. Ligand, receptor, and output files can be specified in any format supported by OpenBabel [5].

7. Output the scores of the top ranked docked poses of each compound.

[1] We performed these dockings using hundreds of cores over a period of weeks. Qualitatively similar results can still be obtained by creating a smaller benchmark by downsampling the number of decoys: `sdsorter -randomize -nbest 200 decoys.sdf decoys_small.sdf`

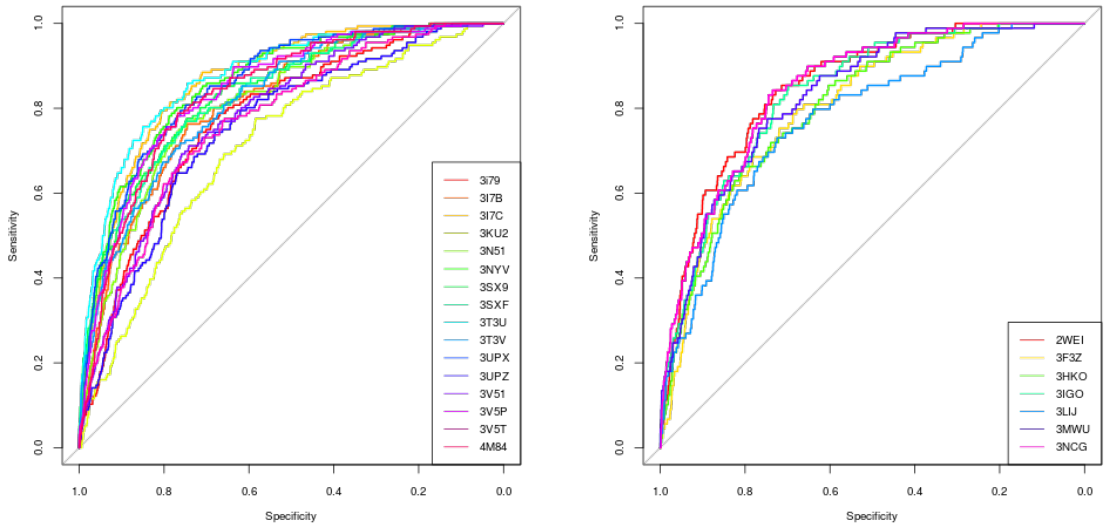
PDB	AUC	Partial AUC
3I79	0.7768	0.5762
3I7B	0.8129	0.6261
3I7C	0.8678	0.6643
3KU2	0.7111	0.5495
3N51	0.8129	0.6152
3NYV	0.8577	0.6779
3SX9	0.8375	0.6742
3SXF	0.8276	0.6641
3T3U	**0.8759**	**0.7091**
3T3V	0.8199	0.6498
3UPX	0.8570	0.6708
3UPZ	0.7544	0.5717
3V51	0.7830	0.5906
3V5P	0.8437	0.6461
3V5T	0.7705	0.5887
4M84	0.8360	0.6268

(a) *T. gondii*

PDB	AUC	Partial AUC
2WEI	0.8520	0.6485
3F3Z	0.8042	0.5983
3HKO	0.8048	0.6106
3IGO	0.8349	0.6291
3LIJ	0.7757	0.5993
3MWU	0.8304	0.6300
3NCG	**0.8491**	**0.6552**

(b) *C. parvum*

Figure 4: Docking performance as measured by AUC and a corrected partial AUC that measures performance at a 10% false positive rate (early enrichment).



(a) *T. gondii* (b) *C. parvum*

Figure 5: Receiver operating characteristic (ROC) curves for docking performance against various CDPK1 structures for each organism.

```
sdsorter -sort minimizedAffinity -reduceconfs 1  2WEI_docked.sdf.gz \
        -print -c > 2WEI_docked.txt
```

8. Analyze the docking performance using R. We will use receiver operating characteristic (ROC) curves, which plot the false positive rate versus the true positive rate as the threshold for acceptance is varied, to assess virtual screening performance. The area under the curve (AUC) of a ROC curve is a measure of accuracy and is equal to the probability that a randomly chosen active compound will be ranked higher than a randomly chosen inactive compound. An AUC of 1.0 is perfect performance, and an AUC of 0.5 is random performance.

   (a) Install and load the pROC package into R.

   ```
   install.packages("pROC")
   library("pROC")
   ```

   (b) Load the scores from docking into R and calculate ROC curves.

   ```
   rarr = list()
   for (file in Sys.glob("*docked.txt")) {
    d = read.table(file,header=T)
    r = roc(grepl("active",d$Title),d$minimizedAffinity,direction=">")
    rarr[[length(rarr)+1]] = r
   }
   names(rarr) = Sys.glob("*docked.txt")
   ```

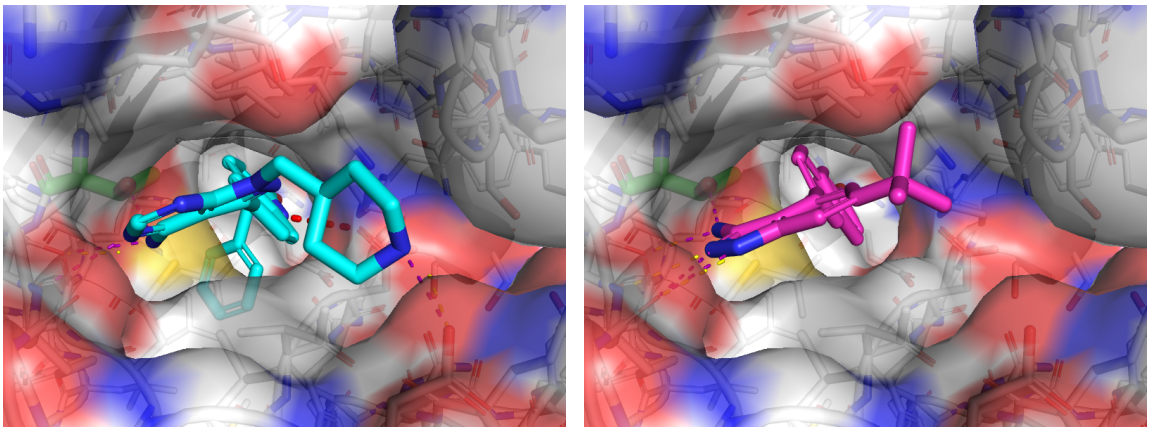
   (c) Print the area under the curve (AUC) and the corrected partial AUC for a 10% false positive rate for each structure. The corrected partial AUC gives the area under the first tenth of the curve and is corrected so that it can be interpreted as a full AUC (e.g., 0.5 is random performance, 1.0 is perfect performance). This is an unbiased, easy to interpret measure of early enrichment.

   ```
   for (n in names(rarr)) {
    r = rarr[[n]]
    a = auc(r,partial.auc=c(1,.9),partial.auc.correct=T)
    s = sprintf("%s %.4f %.4f\n",sub("_docked.txt","",n),r$auc[1],a[1])
    cat(s)
   }
   ```

   The AUCs against all targets are shown in Figure 4 and the ROC curves are shown in Figure 5. The best performing *T. gondii* structure is 3T3U with an AUC of 0.8759 while the best *C. parvum* structure is 3NCG with an AUC of 0.8491.

2 *E. tenella* CDPK1 Model and Test Set Prediction

Here we will create a model of *E. tenella* for virtual screening and develop a custom scoring function based on our previous docking results.



(a) (b)

Figure 6: The docked poses of the top ranked compound when ranked with (a) the default scoring function and with (b) a custom scoring function fit to the data.

2.1 Modeling

Based on the results of the previous section, *T. gondii* is closest to *E. tenella* in overall sequence and in binding site sequence. 3T3U is the *T. gondii* structure that exhibits the best docking performance both in terms of overall performance (AUC) and early enrichment (partial AUC). Therefore we create our model, `swissmodel3T3U.pdb`, using SWISS-Model as in Section 1.1.2.

2.2 Docking of Test Set

Here we dock the provided test set of 22 compounds to our model.

1. Convert to SMILES

```
awk 'NR > 1 {print $2,$1}' tdt2-challenge3-cdpk1_externaltestset.txt \
        > testset.smi
```

2. Generate conformers

```
rdconf.py --maxconfs 1 testset.smi testset.sdf
```

3. Dock the test set

```
smina --autobox_ligand allligs.pdb -r swissmodel3T3U.pdb \
        -l ../testset.sdf -o testset_docked.sdf --seed 0
```

4. Select and sort the best ranked poses

```
sdsorter -sort minimizedAffinity testset_docked.sdf -reduceconfs 1 \
    testset_docked_best.sdf -print -c > testset_default_ranking.txt
```

   The pose of the top ranked compound is shown in Figure 6(a).

```
-0.681459 repulsion(o=0,_c=8)
0.079594  hydrophobic(g=0.5,_b=1.5,_c=8)
-0.073039 non_hydrophobic(g=0.5,_b=1.5,_c=8)
-0.060572 vdw(i=6,_j=12,_s=1,_^=100,_c=8)
0.253043  non_dir_h_bond_lj(o=-0.7,_^=100,_c=8)
2.935961  non_dir_h_bond(g=-0.7,_b=0,_c=8)
-3.472105 acceptor_acceptor_quadratic(o=0,_c=8)
-0.021506 gauss(o=3,_w=2,_c=8)
0.141127  ad4_solvation(d-sigma=3.6,_s/q=0.01097,_c=8)
```

Figure 7: The terms and weights determined by fitting to the docked poses of the *T. gondii* benchmark with logistical regression and backward variable selection.

```
-1.214001 repulsion.o.0._c.8.
3.357897 hydrophobic.g.0.5._b.1.5._c.8.
-3.472107 non_hydrophobic.g.0.5._b.1.5._c.8.
32.238475 vdw.i.6._j.12._s.1._..100._c.8.
-2.465104 non_dir_h_bond_lj.o..0.7._..100._c.8.
3.523563 non_dir_h_bond.g..0.7._b.0._c.8.
-0.573025 acceptor_acceptor_quadratic.o.0._c.8.
-29.073479 gauss.o.3._w.2._c.8.
1.439289 ad4_solvation.d.sigma.3.6._s.q.0.01097._c.8.
```

Figure 8: The average values of each term across the full benchmark after applying the weights of Figure 7.

2.3 Custom scoring with smina

Here we will develop a custom scoring function that is parametrized using the *T. gondii* benchmark. In addition to the default scoring, we will score the held out test with this custom scoring function.

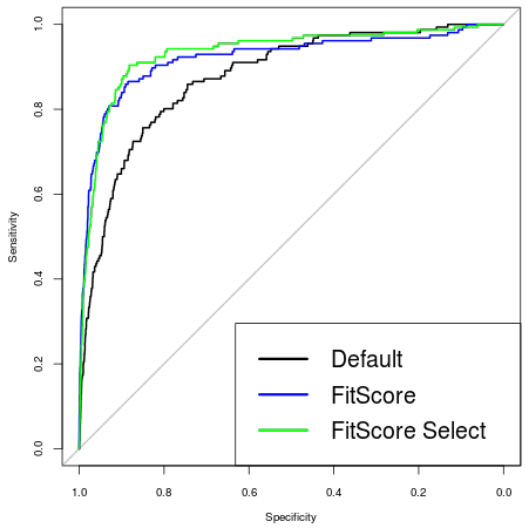
1. Create an custom scoring function with all the conformation dependent terms available in SMINA, omitting the atom-type terms (which would result in a huge number of terms if all possible atom type combinations were considered). Also include the second Gaussian term that is present in the default scoring function. Assign each term a weight of 1.0.

```
smina --print_terms | \
 grep  -v -E 'atom_type|constant|num_|ligand_length' > allterms
echo "gauss(o=3,_w=2,_c=8)" >> allterms
sed -i 's/^/1.0 /' allterms
```

2. Score the top ranked conformations from docking the *T. gondii* benchmark against 3T3U.

```
sdsorter -sort minimizedAffinity -reduceconfs 1 \
        3T3U_docked.sdf.gz 3T3U_docked_single.sdf.gz
smina --custom_scoring allterms --score_only \
        -l 3T3U_docked_single.sdf.gz -r 3T3U.pdb | \
        grep "##" | sed "s/##//" | \
        awk '{print $1 ~ /active/,$0}' > allscores
```

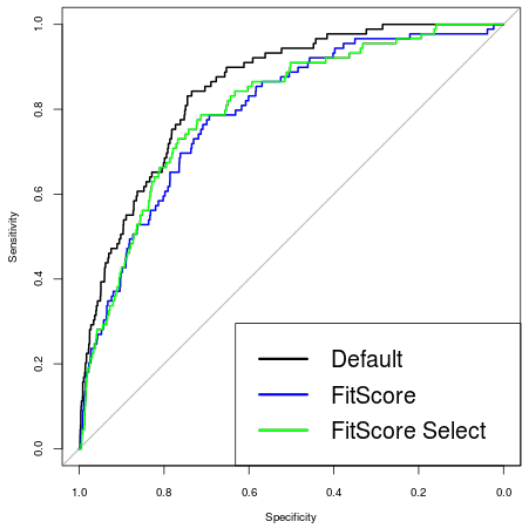
3. Use the `rms` package of R and logistical regression with backward variable selection to fit these scores to the activity data. In the interest of limiting over-fitting to the data, select only those features with a low p-value ($< .0001$).



(a)

	Default	FitScore	FitScore Select
AUC	0.876	0.920	0.933
pAUC10	0.709	0.826	0.816

(b)



(c)

	Default	FitScore	FitScore Select
AUC	0.849	0.793	0.802
pAUC10	0.655	0.611	0.607

(d)

Figure 9: Performance of the scoring function fit to the *T. gondii* benchmark docking data. (a) ROC curves and (b) AUC values when evaluated on the *T. gondii* benchmark. As expected, since the scoring function was parametrized using this data, performance improves. When applied to *C. parvum*, however, the (c) curves and (d) AUCs are not as good for the fit scoring function, suggesting that this scoring function may not generalize well. FitScore Select evaluates the case where the scoring function was used to select the docked pose while FitScore simply re-ranks the poses selected with the default scoring function. In all cases the docked poses were generated using the default scoring function, since the fit scoring function is not parametrized for docking.

```
install.packages("rms")
library(rms)
scores = read.table("allscores",header=T)[c(-2)] #remove Name column
colnames(scores)[1] = "activity"
formula = reformulate(names(scores)[c(-1)],response="activity")
fit = lrm(formula,data=scores,x=T,y=T)
fit2 = fastbw(fit,rule="p",sls=0.0001)
for(n in names(fit2$coefficients)) {
 cat(sprintf("%f %s\n",fit2$coefficients[[n]],n))
}
```

4. Save the determined coefficients to create a new scoring function. Because the term names include characters that are not valid in R names, it is necessary to edit the input to restore the proper terms, resulting in the scoring function shown in Figure 7.

5. Create an average estimate of the contribution of each of the terms in this scoring function by multipling the coefficiencts by the average value of each term.

```
aves = apply(scores[names(coef(fit2))[-1]],2,mean)*coef(fit2)[-1]
for(n in names(aves) ) {
 cat(sprintf("%f %s\n",aves[[n]],n))
}
```

   The resulting averages are shown in Figure 8 and generally conform to expectations. Favorable interactions (hydrophobic, VDW, hydrogen bonds, and solvation) have positive coefficients while unfavorable interactions (non-hydrophobic, repulsion, and acceptor-acceptor) have negative coefficients. Both the steric terms (VDW vs gauss) and the hydrogen bonding terms (linear h-bond vs Lennard Jones h-bond) display some compensation where two terms that measure the same interaction in different ways adopt opposite coefficients to define a new functional.

6. Evaluate this new scoring function. In addition to rescoring the top ranked docked poses, use this new scoring function, `fitscore`, to rescore and re-rank all the docked poses. Note that this scoring function predicts activity rather than affinity. Larger, more positive values indicate compounds that are more likely to be active so compounds need to be sorted in reverse numerical order before choosing the best conformation.

```
smina --score_only -r 3T3U.pdb -l 3T3U_docked_single.sdf.gz \
        --custom_scoring fitscore -o 3T3U_docked_single_rescore.sdf.gz
sdsorter -print -c 3T3U_docked_single_rescore.sdf.gz \
         > 3T3U_docked_single_rescore.txt
smina --score_only -r 3T3U.pdb -l 3T3U_docked.sdf.gz \
        --custom_scoring fitscore -o 3T3U_docked_rescore.sdf.gz
sdsorter -reversesort minimizedAffinity -reduceconfs 1 -print -c \
        3T3U_docked_rescore.sdf.gz > 3T3U_docked_rescore.txt
```

```
rescore = read.table("3T3U_docked_single_rescore.txt",header=T)
rescoreROC = roc(grepl("active",rescore$Title),
        rescore$minimizedAffinity,direction="<")
rescore2 = read.table("3T3U_docked_rescore.txt",header=T)
rescore2ROC = roc(grepl("active",rescore2$Title),
        rescore2$minimizedAffinity,direction="<")
```

   The resulting ROC curves and AUCs are shown in Figure 9. Unsurprisingly, the new scoring function does significantly better when applied to the data it was parametrized on. When applied to the *C. parvum* benchmark (Figure 9), the new scoring function does not perform as well, suggesting it may not generalize to other receptors or even other chemotypes outside of the benchmark data. However, there is a high degree of similarity between *T. gondii* and *E. tenella*, and our *E. tenella* model is based off of the 3T3U structure that was used to parametrize the scoring function. Consequently, it may produce superior performance to the default scoring function when screening against our *E. tenella* model.

7. Score the held out test set with the new scoring function.

```
smina --custom_scoring fitscore --score_only -l testset_docked.sdf \
        -r swissmodel3T3U.pdb -o testset_rescored.sdf
```

```
sdsorter -print -c testset_rescored.sdf -reversesort minimizedAffinity\
        -reduceconfs 1 testset_rescored_best.sdf \
        > testset_custom_ranking.txt
```

The pose of the top ranked compound using this scoring function is shown in Figure 6(b).

The files `testset_custom_ranking.txt` and `testset_default_ranking.txt` are provided at http://bits.csb.pitt.edu/tdtCDPK1 and contain the predicted rank order of the compounds in the held out test set. The scoring function that performs better on this test set should be used to select compounds from the full virtual screening.

3 Virtual Screening

Having demonstrated that docking can be successfully applied to these targets and crafted a custom scoring function tuned to the specific structure our *E. tenella* model is created from, we are ready to screen the eMolecules compound library. Note that our custom scoring function should not be used for docking, as it was parametrized to predict activity from static poses. Consistent with how the scoring function was trained, we will generate all poses using the default scoring function. To achieve this in a reasonable amount of time a large cluster of compute nodes is required.

1. Download eMolecules library.

```
wget http://downloads.emolecules.com/ordersc/2014-01-01/parent.smi.gz
gunzip parent.smi.gz
```

2. Split into appropriately sized pieces for the distributed environment. For example, to split into 10000 chunks:

```
split -n l/10000 parent.smi -d -a 4 split_
for i in split*; do mv $i ${i}.smi; done
```

3. Generate conformers for each piece.

```
rdconf.py --maxconfs 1 split_0000.smi split_0000.sdf.gz
```

4. Dock each piece.

```
smina --autobox_ligand allligs.pdb -r swissmodel3T3U.pdb \
        -l split_0000.sdf.gz -o docked_0000.sdf.gz --seed 0
```

5. Get the top poses from each piece using both default scoring and custom scoring.

```
sdsorter -sort minimizedAffinity -reduceconfs 1 -nbest 100 \
        docked_0000.sdf.gz best_default_0000.sdf.gz
smina --custom_scoring fitscore --score_only -l docked_0000.sdf.gz \
        -r swissmodel3T3U.pdb -o rescored_0000.sdf.gz
sdsorter -reversesort minimizedAffinity -reduceconfs 1 -nbest 100 \
        rescored_0000.sdf.gz best_custom_0000.sdf.gz
```

6. Combine top poses into one file and rank.

```
zcat best_default_*.sdf.gz | gzip > best_default.sdf.gz
zcat best_custom_*.sdf.gz | gzip > best_custom.sdf.gz
sdsorter -sort minimizedAffinity best_default.sdf.gz \
        -nbest 1000 top1000_default.sdf.gz -print -c \
        > top1000_default.txt
sdsorter -reversesort minimizedAffinity best_custom.sdf.gz \
        -nbest 1000 top1000_custom.sdf.gz -print -c \
        > top1000_custom.txt
```

The ranking (`.txt`) and structure files (`.sdf`) of the top 1000 hits from virtual screening are provided at http://bits.csb.pitt.edu/tdtCDPK1/.[2]

Acknowledgments

This work was supported in part by the University of Pittsburgh Center for Simulation and Modeling through the supercomputing resources provided and is funded through R01GM108340 from the National Institute of General Medical Sciences. The content is solely the responsibility of the author and does not necessarily represent the official views of the National Institute of General Medical Sciences or the National Institutes of Health.

References

[1] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 2012. [PubMed:22587354] [PubMed Central:PMC3402020] [doi:10.1021/ci3001277].

[2] Florian Kiefer, Konstantin Arnold, Michael Künzli, Lorenza Bordoli, and Torsten Schwede. The swiss-model repository and associated resources. *Nucleic acids research*, 37(suppl 1):D387–D392, 2009.

[3] David Ryan Koes, Matthew P. Baumgartner, and Carlos J. Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of Chemical Information and Modeling*, 2013.

[4] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet. Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking. *J Med Chem*, 55(14):6582–94, 2012. [PubMed:22716043] [PubMed Central:PMC3405771] [doi:10.1021/jm300687e].

[5] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison. Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, 3:33, 2011. [PubMed:21982300] [PubMed Central:PMC3198950] [doi:10.1186/1758-2946-3-33].

[6] O. Trott and A. J. Olson. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem*, 31(2):455–61, 2010. [PubMed:19499576] [PubMed Central:PMC3041641] [doi:10.1002/jcc.21334].

[2] At the time of submission, the docking of the eMolecules set was only about half complete. The files will be updated within a couple of weeks with the final results.