PROTEIN-LIGAND DOCKING APPLICATION AND COMPARISON USING DISCOVERY STUDIO AND AUTODOCK

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Qi Wang

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Program:
Genomics and Bioinformatics

February 2017

Fargo, North Dakota

North Dakota State University

Graduate School

Title

PROTEIN-LIGAND DOCKING APPLICATION AND COMPARISON USING DISCOVERY STUDIO AND AUTODOCK

By

Qi Wang

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Changhui Yan

Chair

Dr. Yarong Yang

Dr. Jun Kong

Approved:

2/24/2017	Dr. Phillip McClean
Date	Department Chair

ABSTRACT

Protein-ligand docking is a structure-based computational method, which is used to predict the small molecule binding modes and binding affinities with protein receptors. The goals of this study are to compare the docking performances of different software and apply the docking method to predict how protein fatty acid desaturase 1 (FADS1) interact with ligands. Two docking software, Discovery Studio and AutoDock, are used for docking comparison of 195 protein-ligand complexes from PDBind dataset. AutoDock performs a little bit better than Discovery Studio on the docking percentage, which is the percent of the docked complexes out of 195. On the other hand, Discovery Studio has a higher accuracy (successfully docked complexes, within 5 RMSD of the native complex structures) than AutoDock. The interaction between FADS1 and Sesamin shows a similar pattern comparing to the interaction between a homolog of FADS1 and a ligand shown in a PDB structure (PDB id 1EUE).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Protein-ligand interaction is the process of protein interacting with small molecules (referred as ligands) to form stable complexes which have significant biological functions. Protein-ligand complexes play an important role in many biological processes. For example, the serum protein complement factor H (FH) have to interact with some specific glycans on host cell surfaces to function correctly to down-regulate the complement alternative pathway (Blaum, et al., 2015). Thus, a slight change on the structure of glycans might cause serum protein complement factor H to fail on the pathway regulation. Therefore, the accurate protein-ligand interaction modes would be necessary to understand the function of the proteins.

Ligands bind with proteins through intermolecular forces, such as ionic bonds, hydrogen bonds and van der Waals forces. Basically, there are three experimental methods to analyze the structure of protein-ligand complex: X-Ray, Nuclear magnetic resonance spectroscopy (NMR) and electron microscopy. X-Ray crystallography is the most common used experimental technique to study protein-ligand interactions. In general, it involves 7 steps: protein preparation, crystallization, testing crystals, X-ray data collection, structure solution, model building and refinement (Lawson, n.d.). Normally X-Ray crystallography is really time consuming, but the results from it is often reliable and accurate.

Due to the considerable number of publications of protein three-dimensional structures, the protein-ligand docking becomes a hot area recent years. Protein-ligand docking is a structure-based computational method, which is used to predict how small molecules bind with protein receptors and the affinities of the binding. Given the structures of the specific protein and ligand, protein-ligand docking can predict the stable complex using various docking methods and scoring functions. Since protein-ligand docking is a computational method, which only requires the accurate structures of the protein and ligand as the inputs, it can analyze hundreds of interactions simultaneously. Therefore, protein-ligand docking is effective and less time consuming. But on the other hand, the docking results might be influenced by different docking software and scoring functions. To date, there is no docking method that can guarantee perfect binding results. An experimental verification is necessary for any application. Various of protein-ligand algorithms and software are used in biological and pharmaceutical researches, such as disease treatment (Halima, et al., 2016) (Huang, Lee, & Chen, 2014), signal transduction (Khaw, et al., 2014) and drug designs (Dawood, Zarina, & Bano, 2014).

The goals of this study are to compare the docking performances of two docking software, Discovery Studio and AutoDock, and apply the docking method to predict how protein fatty acid desaturase 1 (FADS1) interact with ligands. Discovery Studio is used to predict the 3D structure of FADS1 and its interaction with several ligands. Fatty acid desaturase 1 is an enzyme which can remove the hydrogen atoms from a fatty acid and result in double bonds and the unsaturation of the fatty acid. The protein-ligand docking modes are analyzed between protein FADS1 and the ligands CP-24879, Sesamin, Curcumin, Anthranilicanilide, Dibenzoazepine, Iminodibenzyl, 5H-Dibenz[b,f]azepine, Dibenz[b,f]azepine-5-carbonyl Chloride and Clomipramine Hydrochloride. The interactions are compared with the template interaction between a homolog of FADS1 and a ligand shown in a PDB structure (PDB id 1EUE). The dataset for docking comparison is the PDBbind core set which contains 195 protein-ligand complexes in 65 clusters (Liu, et al., 2014). This dataset can be also widely used as the standard benchmark for evaluating docking and scoring methods.

CHAPTER 2. DOCKING ALGORITHMS AND SCORING FUNCTIONS

In general, protein-ligand docking involves two major steps: complex conformation prediction (docking algorithm) and near-native conformation selection (scoring function). The docking algorithm is aim to use effective methods to find the minimum global energy of protein-ligand complex. The scoring function is used to rank and select the best conformation which ideally should be the same as the natural conformation of the complex.

2.1. Docking methods

Protein-docking involves a large amount of calculation, different algorithms have been developed to predict protein and ligand interactions. Based on their treatment of ligand flexibility, the searching algorithms can be divided into three basic categories: systematic conformational search, stochastic (or random) search and simulation (or deterministic) search.

2.1.1. Systematic conformational search

Systematic protein-ligand docking algorithms allow ligands to rotate in all directions, which often will lead to high cost on future evaluation time. The advantage of this method is that it can evaluate all the possible interactions between protein and ligand. But as the number of combinational evaluations increases, the time to conduct docking increases rapidly. One of the methods to deal with this problem is to define an active site region and let the ligand just rotate within this site, which can greatly reduce the amount of calculation. Another way is to divided the ligand into rigid and flexible fragments. Docking these fragments separately into the active site and then link them together to rebuild the ligand.

DOCK algorithm use anchor-and-grow method to increment conformations. First of all, the ligand is divided into rigid parts, the anchor segments (Meng, Shoichet, & Kuntz, 1992) (Ewinga, Makinoa, Skillmana, & Kuntz, 2001) (Moustakas, et al., 2006). The docking anchor(s) can be selected either by user or some segment size cutoff. Then the anchor is docking to the active site of the protein using geometrical matching. The rest of the ligand can grow freely onto the docked anchor. Finally, local optimization is applied to each conformation.

FlexX algorithm uses MIMUMBA program for conformation generation (Klebe & Mietzner, 1994) (Rarey, Kramer, Lengauer, & Klebe, 1996). Original ligand is separated into different parts and docked into the active site of protein using geometrically restrictive interactions, which mainly based on hydrogen bonds. The bond lengths and angles in the ligand are used as reference for conformations. For each acyclic single bond, it can freely rotate to any preferred torsion angles. Similar to DOCK algorithm, some minimized geometries are used for final optimization.

2.1.2. Stochastic algorithm

The stochastic algorithms randomly change the structure or the position of the ligand. New structure of the ligand is randomly generated and evaluated by some criteria, such as Metropolis or some scoring functions. Monte Carlo method and genetic algorithm are two examples of random algorithm. Some popular software are using stochastic algorithm, such as AutoDock (Goodsell & Olson, 1990), and GOLD (Jones, Willett, Glen, Leach, & Taylor, 1997).

AutoDock algorithm use Lamarckian genetic search for conformation selection (Morris, et al., 2009). Random conformations are created and competing with each other and the conformation with lowest energy is selected and later generations are further created based on the information of current conformation. Other searching methods, such as simulated annealing method and traditional genetic algorithm, can also be used in AutoDock.

A genetic algorithm is used in GOLD software (Jones, Willett, Glen, Leach, & Taylor, 1997) (Jones, Willett, & Glen, 1995) (Verdonk, Cole, Hartshorn, Murray, & Taylor, 2003). In the first stage of docking, parameters for docking are randomized, which include ligand positions in the binding site, ligand rotatable bonds, protein chemical groups and so on. Hydrogen atoms were added to the ligand and the ligand was fully minimized using the MAXIMIN2 module. Then the ligand is docking to the protein and is optimized based on fitting points.

2.1.3. Simulation algorithm

In simulation algorithm, an initial state is determined based on some pre-knowledge of the ligand. And new state is generated based on the previous state. The problem of this method is that some choice of initial state will lead to local minima instead of the real near-native structure. Another issue is that it normally requires high computational cost to get the potential protein-ligand complex structure. Molecular dynamics and energy minimization are two widely used simulation methods. There are some standardized packages for molecular dynamic, for example CHARMM (Brooks, et al., 2009), Amber and GROMACS. But unlike molecular dynamics, energy minimization method is barely used alone but combined with some other searching algorithms.

CHARMM is a program for molecular simulation and modeling (Brooks, et al., 2009). It uses energy minimization techniques to optimize the conformations, performs molecular dynamics simulation, and analyzes the simulation results to determine structural, equilibrium, and dynamic properties.

2.1.4. Receptor flexibility

Since receptor proteins are much more complex than ligands, protein with full flexibility during docking procedure would increase calculation complexity dramatically. But some degrees of receptor flexibility are available in a lot of software. Most approaches of receptor flexibility would apply some restrictions on the protein, for example some software requires an active site and allows the amino acids within the active site rotate freely, some would divide the protein into rigid part and flexible part to reduce the calculation time. Similar algorithms applied to ligand flexibility could also be used to analyze receptor flexibility, such as Monte Carlo method (Trosset & Scheraga, 1999) and molecular dynamics (Pak & Wang, 2000).

2.2. Scoring functions

After docking, multiple conformations of protein-ligand docking complexes are generated using various algorithms. Next step would be to evaluate and rank the conformations based on scoring functions. Because thousands of conformations might be generated from docking procedure, scoring and ranking all the conformations are time consuming. The key function of scoring procedure is to effectively differentiate the near-native complexes form incorrect ones. Currently a number of different scoring functions are available, which can be divided into three types: force-field-based, empirical and knowledge-based scoring functions.

2.2.1. The force-field based scoring function

The force-field-based scoring function can evaluate the potential energy of a system, as the sum of different particles (ligand and protein) in the system. Normally, the receptor-ligand interaction energy and internal ligand energy are evaluated using the force-field-based scoring function and most solvent effects as well as solute entropies are ignored. Coulomb and van der Waals interactions are often used in the scoring functions to calculate the energy (Goodsell & Olson, 1990) (Meng, Shoichet, & Kuntz, 1992).

AMBER force field is a widely-used scoring function to calculate the total binding energy of protein-ligand docking (Cornel, et al., 1995).

2.2.2. Empirical scoring function

Empirical methods use physical-chemical properties of known protein-ligand complexes to predict the free binding energy of a predicted conformation. Empirical methods are usually less computational demanding than force-field-based methods.

Hans-Joachim Bohm (Bohm, 1994) developed an empirical scoring function to calculate the free energy of binding for protein-ligand complexes. This function includes the hydrogen bonds, ionic interactions, the lipophilic protein-ligand contact surface and the number of rotatable bonds in the ligand.

$$\Delta G_{binding} = \Delta G_0 + \Delta G_{hb} \sum_{h-bonds} f(\Delta R, \Delta\alpha) + \Delta G_{ionic} \sum_{ionic-int} f(\Delta R, \Delta\alpha) + \Delta G_{lipo} |A_{lipo}| + \Delta G_{rot} NROT$$

$$f(\Delta R, \Delta\alpha) = f1(\Delta R) f2(\Delta\alpha)$$

where $f(\Delta R, \Delta\alpha)$ is a penalty function related with hydrogen-bond length and angle. The problem of this function is that it does not take into account the water-mediated hydrogen bonds, which might take an important role in protein-ligand binding. And obviously the accuracy of this scoring function highly depends on the experimental binding energies, which might not available sometime.

2.2.3. Knowledge-based scoring function

Knowledge-based scoring functions use the frequency of experimental structures in large 3D databases to evaluate the possibility of the protein-ligand complex. Not like empirical methods, knowledge-based methods do not need any additional analysis on the training dataset, which reduces the amount of calculation. But on the other hand, it is also limited by the size of the database used.

CHAPTER 3. CASE STUDY

To analyze the docking performances, protein FADS1 was used to study the binding modes with 9 ligands: CP-24879, Sesamin, Curcumin, Anthranilicanilide, Dibenzoazepine, Iminodibenzyl, 5H-Dibenz[b,f]azepine, Dibenz[b,f]azepine-5-carbonyl Chloride and Clomipramine Hydrochloride. Furthermore, the PDBbind core set containing 195 protein-ligand complexes was used to compare the docking results of different software, Discovery Studio and AutoDock.

3.1. PDBbind data set

The PDBbind core set contains 195 protein-ligand complexes in 65 clusters (Liu, et al., 2014), which is a part of the PDBbind dataset, which includes a collection of the bimolecular complexes binding affinity measured with experiments in the Protein Data Bank (PDB). Each cluster in the dataset is selected by the protein sequence similarity with 90% cutoff and it contains 3 members: the one with the highest, medium and the lowest binding constant ($\log K_a$). The PDBbind core set is a high-quality benchmark for evaluating different docking methods and scoring functions. A study of the docking performances has been done among Discovery Studio 3.5, GOLD 5.1, SYBYL 8.1 Schrodinger 2011, MOE 2011 Academic software 1.3 (Li, Han, Liu, & Wang, 2014). One the other hand, AutoDock is the most highly used docking software lately (Sousa, et al., 2013). Therefore, the two software, Discovery Studio 4.1 and AutoDock 4.0, are selected for the docking comparison. For each protein-ligand complex in PDBbind core set, the resolution of the structure is smaller than 2.5 A and the inhibition constant (Ki,) or dissociation constants (Kd) is known. In X-Ray crystallography, resolution is the highest value in the diffraction pattern (Frank, 2006). And the smaller the resolution is, the less errors in the structures (Huang Y.-F. , 2007). Ki and Kd are special types of equilibrium constants that are theoretical relative to each other. This dataset can be used as the standard benchmark for evaluating docking and scoring methods.

3.2. Protein FADS1

The protein FADS1 is the fatty acid desaturase 1 enzyme in Human, which is located in chromosome 11q12.2-13.1 (Nakamura & Nara, 2004). The fatty acid chain is the foundation of biological membranes and the degree of unsaturation would highly influence the melting temperature and the fluidity of the membranes. Fatty acid desaturase 1 can remove the hydrogen atoms from a fatty acid and result in double bonds and the unsaturation of the fatty acid. It plays an important role in lipid metabolic pathway. The ligands used in this study are CP-24879, Sesamin, Curcumin, Anthranilicanilide, Dibenzoazepine, Iminodibenzyl, 5H-Dibenz[b,f]azepine, Dibenz[b,f]azepine-5-carbonyl Chloride and Clomipramine Hydrochloride. The docking between FADS1 and the ligands will provide another way to better understand the function of fatty acid desaturase 1. The sequence of the protein can be obtained on *UniProt.org* (UniProtKB - O60427 (FADS1_HUMAN), 2017). It is 444 amino acids long and its 3D structure is still unknown.

```
>sp|FADS1|1-444
MAPDPVAAETAAQGPTPRYFTWDEVAQRSGCEERWLVIDRKVYNISEFTRRHPGGSRVIS
HYAGQDATDPFVAFHINKGLVKKYMNSLLIGELSPEQPSFEPTKNKELTDEFRELRATVE
RMGLMKANHVFFLLYLLHILLLDGAAWLTLWVFGTSFLPFLLCAVLLSAVQAQAGWLQHD
FGHLSVFSTSKWNHLLHHFVIGHLKGAPASWWNHMHFQHHAKPNCFRKDPDINMHPFFFA
```

LGKILSVELGKQKKKYMPYNHQHKYFFLIGPPALLPLYFQWYIFYFVIQRKKWVDLAWMI TFYVRFFLTYVPLLGLKAFLGLFFIVRFLESNWFVWVTQMNHIPMHIDHDRNMDWVSTQL QATCNVHKSAFNDWFSGHLNFQIEHHLFPTMPRHNYHKVAPLVQSLCAKHGIEYQSKPLL SAFADIIHSLKESGQLWLDAYLHQ

CHAPTER 4. METHODS

4.1. LibDock (Discovery Studio)

LibDock uses the systematic conformational search algorithm to dock ligands freely to the receptor and rank the compounds via the default scoring function LigScore (Krammer, Kirchhoff, Jiang, Venkatachalam, & Waldman, 2005). First, random conformations of each ligand from 195 protein-ligand complexes with different rotatable single non-ring bonds were generated to calculate the internal energy by using van der Waals potentials and a dihedral angle term. The conformations will be minimized using Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm (Fletcher, 1987) and ranked based on SASA, which is the solvent accessible surface area of a specific conformation. Then the binding sites were determined by locating the apolar and polar hot spots on the protein. The hot spots are the locations within the binding sphere that have a high chance to form either an apolar bond or a hydrogen bond. Thirdly, the geometric hashing algorithm was used to dock the conformations to the binding site of protein. Finally, the complexes were optimized using BFGS optimization algorithm, ranked and clustered for in the final stage (Diller & Merz, 2001).

All the proteins and ligands have been standardized by applying the CHARMm forcefield to the proteins and monitoring the valences of the ligands. After the preparation, a sphere was defined around the binding site for each protein. The spheres are defined by randomly selecting about 10 amino acids around the native binding site of the protein to define it. The binding site sphere is a required input for running LibDock in Discovery Studio. The number of polar or apolar receptor binding site features (hotspots) was 200, which is chosen to increase the chance of finding the native protein-ligand structure while still has a reasonable computational time. To ensure the docking quality, the RMSD tolerance (Å) was chosen as 1 Å.

4.2. Autodock

Autodock uses the stochastic algorithm to optimize the random conformations with the lowest energy. At first, the protein receptor is embedded in a grid with 40 grid points in each of the x-y-z direction centering (15.45, 26.233, 3.593). The grid spacing is 0.375 Å. Then, the ligand can be put at each grid point with a random initial position and Dihedral offset. A receptor-ligand interaction energy calculated and stored using the formula:

$$\Delta \mathrm{G} = \Delta G_{vdw} + \Delta G_{hbond} + \Delta G_{elec} + \Delta G_{cov} + \Delta G_{tor} + \Delta G_{sol}$$

where ΔG_{vdw} stands for the energy for van der Waals, ΔG_{hbond} represents hydrogen bond, ΔG_{elec} is electrostatics, ΔG_{cov} measures the deviations from covalent geometry, ΔG_{tor} models the internal and external rotation restriction and ΔG_{sol} models the solvent entropy changes (Morris, et al., 1998). Also each conformation of the ligand generated by Monte Carlo simulated annealing search is allowed to search its local space in the current valley by replacing the conformation based on the result to find the minima, which can be used in the later generation (Morris, et al., 2009).

In Autodock, formatted ligand files are required in pdbqt format, which contain atom types as well as rotatable bonds supported by AutoDock. Protein and ligand files are prepared using the Python scripts provided by AutoDock. For the docking procedures, the initial position of ligand and relative dihedral offset set to be random. Genetic algorithm (GA) is used to search parameters, such as number of GA runs, maximum number of evaluations, rate of nutation and so on, with all default parameters. Defaults are also used in the docking parameters for random number generator, energy parameters, step size parameters and output format parameters. After that, .dpf files are saved containing docking parameters and instructions for Lamarakian Genetic Algorithm docking (Morris, et al., 1998), which is also known as Genetic Algorithm Local Search. Finally, with all parameters set, the .dpf files are required to run AutoDock. All the docking results are clustered using a tolerance of 3.0 Å. For each protein-ligand complex, 10 generations of Genetic algorithm have been run with 50 cycles in each run and the maximum number of conformations in each cycle is set to be 25000.

4.3. Protein FADS1

The protein FADS1 is the fatty acid desaturase 1 protein in Human. Since the 3D structure of this protein is still unknown, the first step is to predict the 3D structure of FADS1. Currently there are two major methods for protein structure prediction: template-based modeling and free modeling (Zhang, 2008). The template-based modeling, also known as homology modeling, is to predict the structure using the known structures of the templates who share similar sequences with the target protein. The result of homology modeling is highly depending on the template alignment and selection. And it is possible to build high quality models given close templates. Free modeling, also termed as "de novo" modeling, is mainly using physical principles or sometimes small fragments to build the 3D structure of the target protein. But this approach is often time consuming and the prediction qualities for large proteins are usually poor. In this study, homology modeling is used to study the interaction between FADS1 and its possible ligands.

For templates alignment and selection, the Basic Local Alignment Search Tool (BLAST) within Discovery Studio is used with E-value cutoff equals to 10 in the PDB_nr95 database. The scoring matrix of this search is BLOSUM62 with the word size 3. The gap existence penalty is 11 and gap extension penalty is 1. Based on the Identity, alignment length, Resolution, E-value and the Organism of the structures, 6 homology proteins are selected as the templates to build the 3D structure of FADS1: 1EUE, 1LJ0, 1CYO, 2M33, 3NER and 2I96.

Table 1

Templates alignment results

PDB ID	Identity with FADS1	Alignment Length	Resolution	E-value	Organism
1EUE_B	43	57	1.8	5.278 e-11	*Rattus norvegicus*
1LJ0_A	42	57	2	1.079 e-10	*Rattus norvegicus*
1CYO_A	31	82	1.5	6.014 e-10	*Bos taurus*
2M33_A	31	82		9.042 e-10	*Oryctolagus cunic*
3NER_B	43	53	1.45	1.240 e-09	*Homo sapiens*
2I96_A	31	89		1.615 e-09	*Homo sapiens*

The possible ligands of protein FADS1 are CP-24879, Sesamin, Curcumin, Anthranilicanilide, Dibenzoazepine, Iminodibenzyl, 5H-Dibenz[b,f]azepine, Dibenz[b,f]azepine-5-carbonyl Chloride and Clomipramine Hydrochloride in this study. (Structures of the ligands are showd in Appendix A.) For docking preparation, the FADS1 protein and all 9 ligands have been standardized by applying the CHARMm (Chemistry at Harvard Macromolecular Mechanics) forcefield, which uses some formula and parameters to calculate the potential energy of a system. Also the valences of the ligands need to be balanced for correct docking. After the preparation, a sphere was defined around the binding site the receptor protein, which covers the entire FADS1 protein. A binding site sphere is required for LibDock in Discovery Studio. To increase the possible conformations, the number of polar or apolar receptor binding site features (hotspots) was 200 and the RMSD tolerance was chosen as 1 Å. The root mean square deviation (RMSD) is a measurement of the average atom distance between two molecules, which is calculated using the formula:

$$\mathrm{RMSD(a,b)} = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}\left[(a_{ix}-b_{ix})^2+\left(a_{iy}-b_{iy}\right)^2+(a_{iz}-b_{iz})^2\right]}$$

where *i* refers to the atoms in molecules *a* and *b*, *n* is the total number of atoms and *x*, *y*, *z* are the *x*-*y*-*z* coordinates in three-dimensional space. Therefore, the smaller RMSD it, the closer the protein-ligand complex is to the native structure.

Docking preferences was set to be High quality, which is a specific mode in Discovery Studio with all parameters are predefines. The conformation method was FAST, which quickly generate diverse low-energy conformations using a systemic search for small molecules. To reduce the time consumption, no minimization method was used in all the docking processes. Other parameters, such as sp2-sp2 rotation grid scoring, were kept on default settings (true).

CHAPTER 5. RESULTS

5.1. PDBbind Dataset

The results of the docking software evaluation are summarized in Table 1. The successfully docked complexes are considered to be within 3.0 Å tolerance of RMSD. A larger RMSD tolerance will increase the successfully docking percentage. But the protein-ligand complexes with larger RMSD are less reliable than the ones with smaller RMSD. The successfully docking percentage is defined as the percentage of the docked complexes having a RMSD less than or equal to 3.0 Å among 195 protein-ligand complexes. Figure 1 and 2 show the protein-ligand docking RMSD summary of Discovery Studio and AutoDock. It is clear that the predicted complex RMSD using Discovery Studio is more stable, mainly around 10 Å, comparing to the complex RMSD using AutoDock, which has a higher percentage on the RMSD greater than 15 Å. AutoDock performs a little bit better than Discovery Studio regarding to the successfully docking percentage, 16.92% (33 out of 195) and 10.26% (20 out of 195), respectively. But while comparing the minimum RMSD for the two software, Discovery Studio has 109 protein-ligand complexes with lower RMSD than their results of AutoDock. Detailed docking results from both softwares are showed in Appendix C.

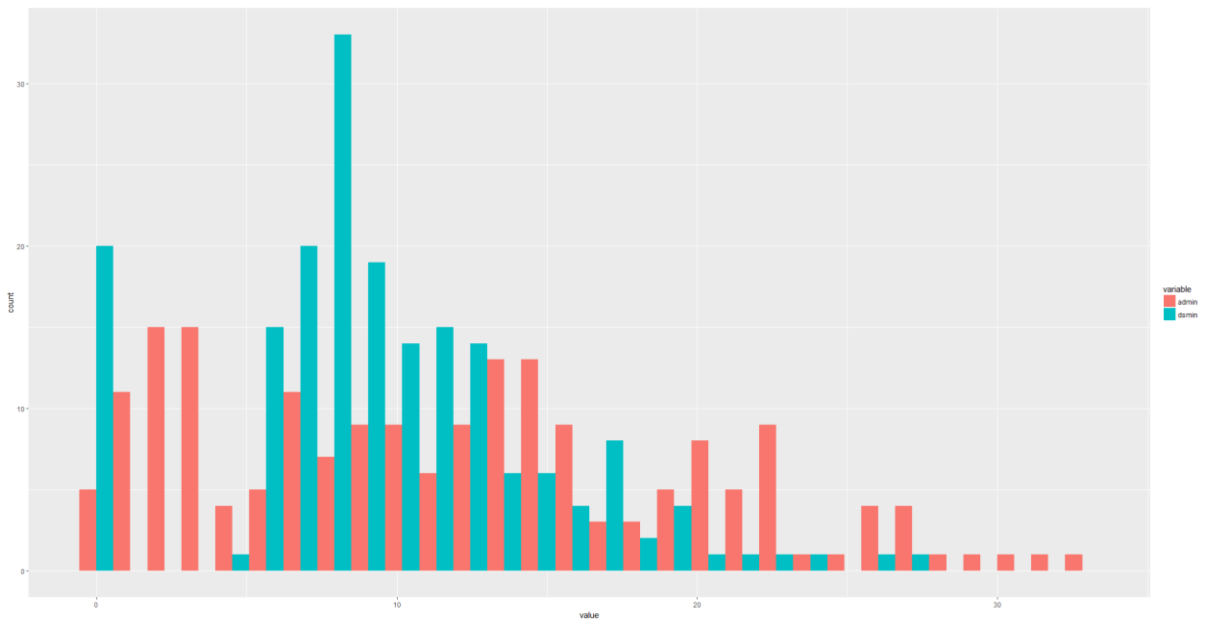


Figure 1. The histograms of RMSD for Discovery Studio and AutoDock results

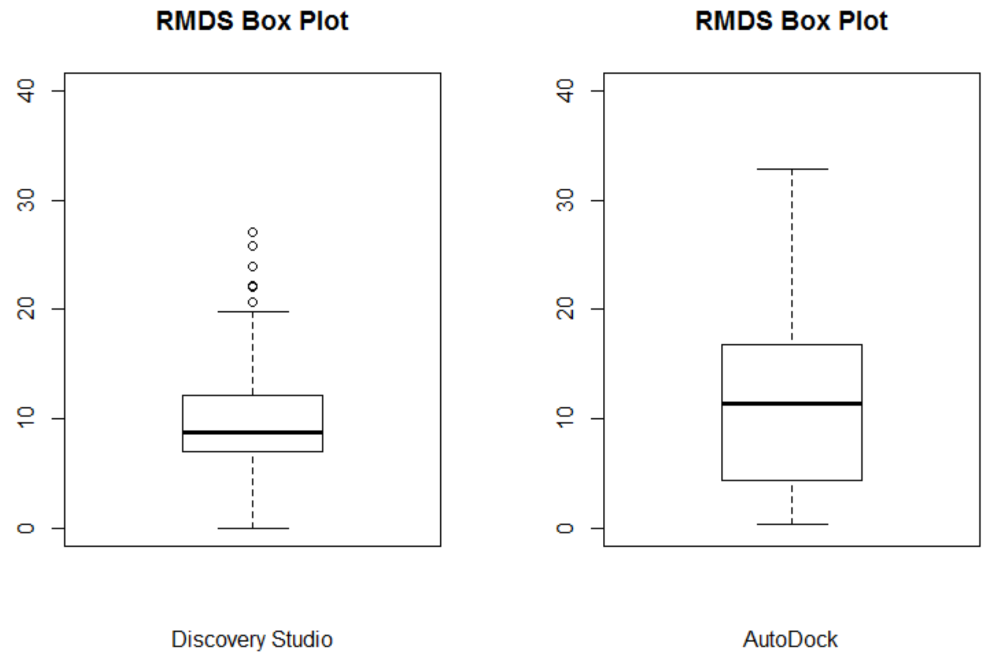
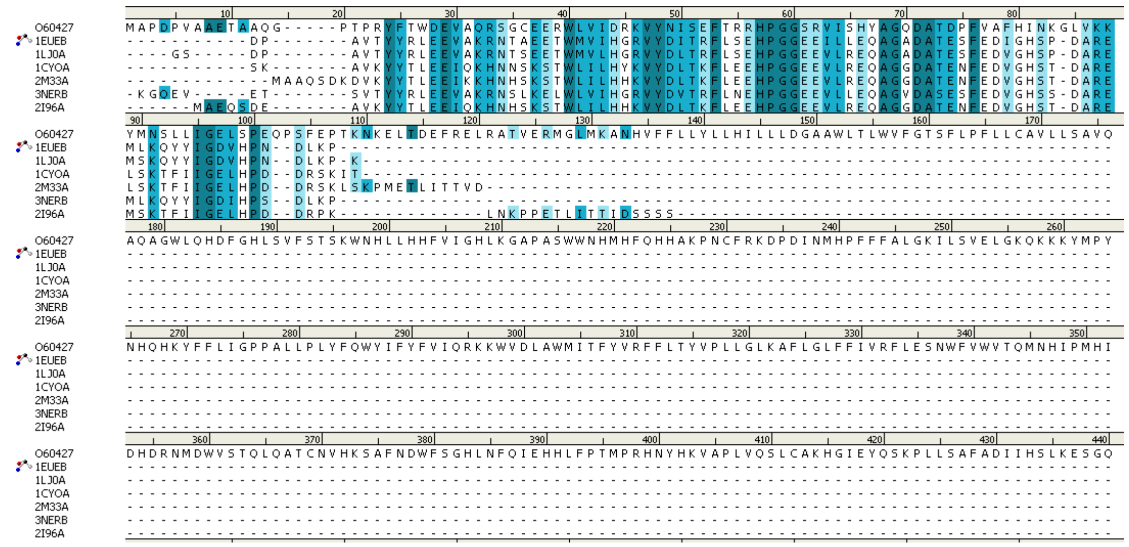


Figure 2. The box-plot of the RMSD values

5.2. Protein FADS1

Based on the Identity, alignment length, Resolution, E-value and the Organism of the structures, 6 homology sequences are selected as the templates to build the 3D structure of FADS1: 1EUE, 1LJ0, 1CYO, 2M33, 3NER and 2I96. Figure 3 shows the protein FADS1 alignment with 6 Homology sequences from BLAST search. The sequences in blue color are highly conserved, which is good for predicting the 3D structure of FADS1 through alignment. One thing needs to be mention that there is no sequence alignment beyond amino acid 138 L to the last amino acid 440 Q, thus no reliable 3D structure could possible generated for this part of the sequence.



```
O60427  MAPDPVAAETAAQG-----PTPRYFTWDEVAQRSGCEERWLVIDRKVYNISEFTRRHPGGSRVISHYAGQDATDPFVAFHINKGLVKK
1EUEB   -----------DP-------AVTYYRLEEVAKRNTAEETWMVIHGRVYDITRFLSEHPGGEEILLEQAGADATESFEDIGHSP-DARE
1LJ0A   ----GS-----DP-------AVTYYRLEEVAKRNTSEETWMVLHGRVYDLTRFLSEHPGGEEVLREQAGADATESFEDVGHSP-DARE
1CYOA   -----------SK-------AVKYYTLEEIQKHNNSKSTWLILHYKVYDLTKFLEEHPGGEEVLREQAGGDATENFEDVGHST-DARE
2M33A   -------------MAAQSDKDVKYYTLEEIKKHNHSKSTWLILHHKVYDLTKFLEEHPGGEEVLREQAGGDATENFEDVGHST-DARE
3NERB   -KGQEV-----ET-------SVTYYRLEEVAKRNSLKELWLVIHGRVYDVTRFLNEHPGGEEVLLEQAGVDASESFEDVGHSS-DARE
2I96A   ------MAEQSDE-------AVKYYTLEEIQKHNHSKSTWLILHHKVYDLTKFLEEHPGGEEVLREQAGGDATENFEDVGHST-DARE

O60427  YMNSLLIGELSPEQPSFEPTKNKELTDEFRELRATVERMGLMKANHVFFLLYLLHILLLDGAAWLTLWVFGTSFLPFLLCAVLLSAVQ
1EUEB   MLKQYYIGDVHPN--DLKP---------------------------------------------------------------------
1LJ0A   MSKQYYIGDVHPN--DLKP-K-------------------------------------------------------------------
1CYOA   LSKTFIIGELHPD--DRSKIT-------------------------------------------------------------------
2M33A   LSKTFIIGELHPD--DRSKLSKPMETLITTVD--------------------------------------------------------
3NERB   MLKQYYIGDIHPS--DLKP---------------------------------------------------------------------
2I96A   MSKTFIIGELHPD--DRPK-------------LNKPPETLITTIDSSSS---------------------------------------

O60427  AQAGWLQHDFGHLSVFSTSKWNHLLHHFVIGHLKGAPASWWNHMHFQHHAKPNCFRKDPDINMHPFFFALGKILSVELGKQKKKYMPY
1EUEB   ----------------------------------------------------------------------------------------
1LJ0A   ----------------------------------------------------------------------------------------
1CYOA   ----------------------------------------------------------------------------------------
2M33A   ----------------------------------------------------------------------------------------
3NERB   ----------------------------------------------------------------------------------------
2I96A   ----------------------------------------------------------------------------------------

O60427  NHQHKYFFLIGPPALLPLYFQWYIFYFVIQRKKWVDLAWMITFYVRFFLTYVPLLGLKAFLGLFFIVRFLESNWFVWVTQMNHIPMHI
1EUEB   ----------------------------------------------------------------------------------------
1LJ0A   ----------------------------------------------------------------------------------------
1CYOA   ----------------------------------------------------------------------------------------
2M33A   ----------------------------------------------------------------------------------------
3NERB   ----------------------------------------------------------------------------------------
2I96A   ----------------------------------------------------------------------------------------

O60427  DHDRNMDWVSTQLQATCNVHKSAFNDWFSGHLNFQIEHHLFPTMPRHNYHKVAPLVQSLCAKHGIEYQSKPLLSAFADIIHSLKESGQ
1EUEB   ----------------------------------------------------------------------------------------
1LJ0A   ----------------------------------------------------------------------------------------
1CYOA   ----------------------------------------------------------------------------------------
2M33A   ----------------------------------------------------------------------------------------
3NERB   ----------------------------------------------------------------------------------------
2I96A   ----------------------------------------------------------------------------------------
```

Figure 3. The sequences alignment of FADS1 with templates.

Figure 4 is the predicted 3D structure of FADS1 based on the structures of the homology sequences. This protein folds a β-sheet (in blue color) in the middle surrounded by several α-helices (in red color). Thus a hydrophobic binding site is formed in the center.

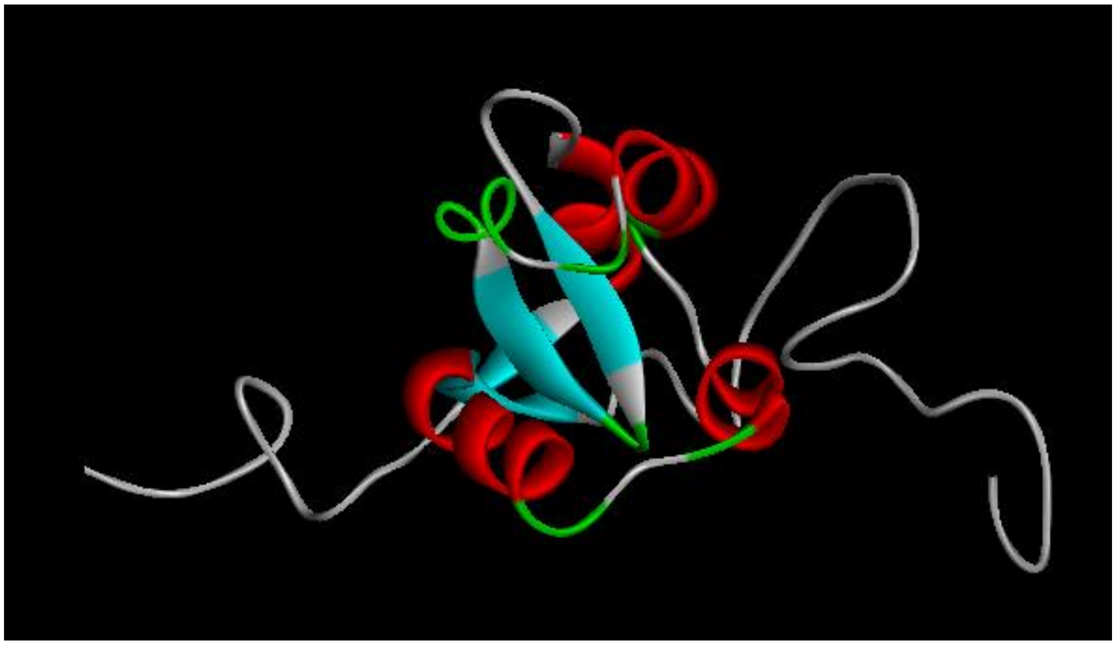


Figure 4. The predicted 3D structure of FADS1

By comparing the docking results between FADS1 with 9 ligands and the template 1EUE with Protoporphyrin IX containing Fe, it showed that the interaction between FADS1 and Sesamin has the highest similarity to the template complex. 1EUE is rat outer mitochondrial membrane cytochrome B5 protein, which belongs to the electron transport system (Oganesyan & Zhang, 2001). The amino acids ILE45, LEU46, ALA54, PHE58 and ALA67 in 1EUE are important in the interaction with Protoporphyrin IX. The detailed information of the interactions is showed in Table 2.

Table 2

Interaction between 1EUE and Protoporphyrin IX

Amino acid	Category	types	Distance
ILE45	Hydrophobic	Alkyl	3.643370
LEU46	Hydrophobic	Alkyl	5.214600
ALA54	Hydrophobic	Alkyl	3.777066
PHE58	Hydrophobic	Pi-Alkyl	4.409321
ALA67	Hydrophobic	Alkyl	3.676638

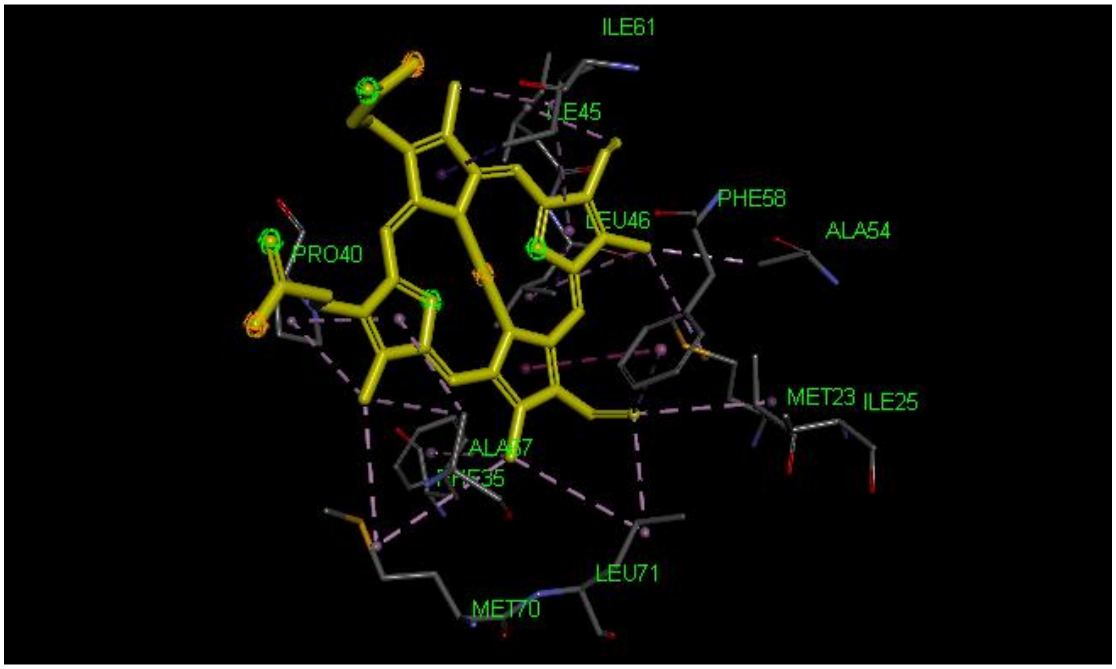


Figure 5. 1EUE chain B interacting with Protoporphyrin IX containing Fe

Based on the results of alignment, it's clear that VAL94, ILE95, ALA103, PHE107 and VAL117 are the sequence aligned amino acids in FADS1, which also play important roles in the interaction with Sesamin. Figure 5 and 6 shows the interaction results. The results indicate that the interaction between FADS1 and Sesamin share a similar binding pattern to the interaction between 1EUE and Protoporphyrin IX. Thus it will help us better understand the biological function of FADS1 as well as shed some light on drug design.

Table 3

Interaction between FADS1 and Sesamin

Amino acid	Category	types	Distance
VAL94	Hydrophobic	Pi-Alkyl	5.368634
ILE95	Hydrophobic	Alkyl	5.476260
ALA103	Hydrophobic	Pi-Alkyl	4.461106
PHE107	Hydrophobic	Pi-Alkyl	5.171723
VAL117	Hydrophobic	Pi-Alkyl	5.087928

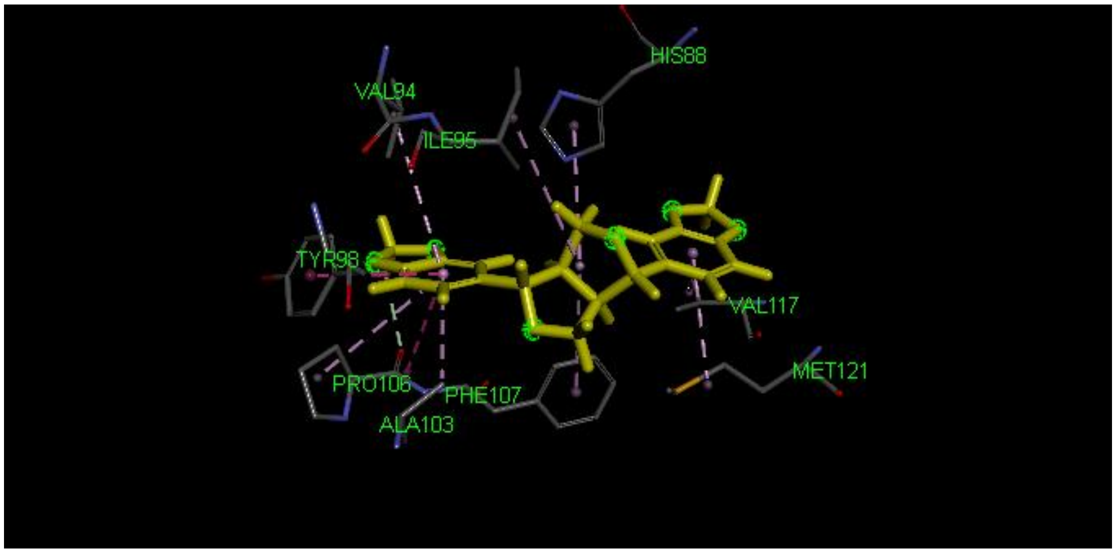


Figure 6. FADS1 interacting with sesamin

CHAPTER 6. DISCUSSION

The goals of this study are to compare the docking performances of different software and apply the docking method to predict how protein fatty acid desaturase 1 (FADS1) interact with ligands. Two docking software, Discovery Studio and AutoDock, are used for docking comparison of 195 protein-ligand complexes from PDBind dataset. The PDBbind core set is widely used as the standard benchmark for evaluating docking and scoring methods. The docking results show that the predicted complex RMSD using Discovery Studio is more stable, mainly around 10 Å, comparing to the complex RMSD using AutoDock, which has a higher percentage on the RMSD greater than 15 Å. AutoDock performs a little bit better than Discovery Studio regarding to the successfully docking percentage, 16.92% (33 out of 195) and 10.26% (20 out of 195), respectively. But while comparing the minimum RMSD gained by the two softwares, Discovery Studio has 109 protein-ligand complexes with lower RMSD than their results of AutoDock. The docking accuracy of protein-ligand complexes is highly related with the specific complexes as well as the docking software. Some complexes could not be successfully docked based on the specific parameter settings using one software, but can get somewhat accurate result using the other one. All the results are run based on the default settings; therefore it's possible to get a higher accuracy for specific complex by trying different combinations of parameters.

Discovery Studio is commercial software and the installation cost of it is pretty high comparing to the free of charged AutoDock. But Discovery Studio provides detailed tutorials for users to get familiar with its functions and the technical support team from the Accelrys

Company is very helpful with troubleshooting of Discovery Studio. On the other hand, limited tutorials are given in the AutoDock website regarding docking using AutoDock. Also the understanding of Python language is pretty useful while dealing with hundreds of protein-ligand docking using the same parameter settings.

Discovery Studio is used to predict the 3D structure of protein fatty acid desaturase 1 (FADS1) and its interaction with several ligands. Fatty acid desaturase 1 is an enzyme which can remove the hydrogen atoms from a fatty acid and result in double bonds and the unsaturation of the fatty acid. It plays an important role in lipid metabolic pathway. The 3D structure of FADS1 is predicted using homology modeling based on its amino acid sequence. Based on the Identity, alignment length, Resolution, E-value and the Organism of the structures, 6 homology sequences (1EUE, 1LJ0, 1CYO, 2M33, 3NER and 2I96) are selected as the templates to build the 3D structure of FADS1. The 9 of its possible ligands for FADS1 are CP-24879, Sesamin, Curcumin, Anthranilicanilide, Dibenzoazepine, Iminodibenzyl, 5H-Dibenz[b,f]azepine, Dibenz[b,f]azepine-5-carbonyl Chloride and Clomipramine Hydrochloride. As a result of the docking, the interaction between FADS1 and Sesamin shows a similar pattern comparing to the interaction between a homolog of FADS1 and a ligand shown in a PDB structure (PDB id 1EUE). The structures of the other 8 protein-ligand complexes of FADS1 are not as close to the template structure as FADS1-Sesamin complex. The interaction between FADS1 and Sesamin would provide another way to understand the function of fatty acid desaturase 1 and possible drug design.

REFERENCES

Blaum, B. S., Hannan, J. P., Herbert, A. P., Kavanagh, D., Uhrín, D., & Stehle, T. (2015). Structural basis for sialic acid–mediated self-recognition by complement factor H. *Nature Chemical Biology, 11*, 77–82. doi:10.1038/nchembio.1696

Bohm, H.-J. (1994). The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. *Journal of Computer-Aided Molecular Design, 8*, 243-256.

Brooks, B. R., Brooks, C. L., MacKerell, A. D., Nilsson, L., Petrella, R. J., Roux, B., . . . Karplus, M. (2009). CHARMM: The Biomolecular Simulation Program. *Journal of Computational Chemistry, 30*(10), 1545-1614.

Cornel, W. D., Cieplak, P., Bayly, C. I., Gould, I. R., Merz, K. M., Ferguson, D. M., . . . Kollman, P. A. (1995). A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *Journal of the American Chemical Society*, 5179-5197.

Dawood, S., Zarina, S., & Bano, S. (2014). Docking studies of antidepressants against single crystal structure of tryptophan 2, 3-dioxygenase using Molegro Virtual Docker software. *Pakistan journal of pharmaceutical sciences, 27*(5), 1529-1539.

Diller, D. J., & Merz, K. M. (2001). High Throughput Docking for Library Design and Library Prioritization. *PROTEINS: Structure, Function, and Genetics, 43*, 113-124.

Ewinga, T. J., Makinoa, S., Skillmana, G., & Kuntz, I. D. (2001). DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases. *Journal of Computer-Aided Molecular Design*, 411-428.

Fletcher, R. (1987). *Practical methods of optimization.* New York: John Wiley & Sons.

Frank, J. (2006). *Three-Dimnsional Electron Microscopy of Macromolecular Assemblies.* New York: Oxford University Press.

Goodsell, D. S., & Olson, A. J. (1990). Automated docking of substrates to proteins by simulated annealing. *Proteins Structure Function and Bioinformatics, 8*(3), 195-202. doi:10.1002/prot.340080302

Halima, S. B., Mishra, S., Raja, K. P., Willem, M., Baici, A., Simons, K., . . . Rajendran, L. (2016). Specific Inhibition of β-Secretase Processing of the Alzheimer Disease Amyloid Precursor Protein. *Cell Reports, 14*(9), 2127-2141. doi:10.1016/j.celrep.2016.01.076

Huang, H.-J., Lee, C.-C., & Chen, C. Y.-C. (2014). Medicine, Lead Discovery for Alzheimer's Disease Related Target Protein RbAp48 from Traditional Chinese. *BioMed Research International*, Article ID 764946. doi:doi:10.1155/2014/764946

Huang, Y.-F. (2007, 12 11). *Study of Mining Protein Structural Properties and its Application* . Retrieved from http://www.csie.ntu.edu.tw/~yfhuang/papers/phdprop.yfhuang.pdf

Jones, G., Willett, P., & Glen, R. C. (1995). Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *Journal of Molecular Biology, 245*(1), 43-53. doi:10.1016/S0022-2836(95)80037-9

Jones, G., Willett, P., Glen, R. C., Leach, A. R., & Taylor, R. (1997). Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology, 267*(3), 727-748. doi:http://dx.doi.org/10.1006/jmbi.1996.0897

Khaw, K. Y., Choi, S. B., Tan, S. C., Wahab, H. A., Chan, K. L., & Murugaiyah, V. (2014). Prenylated xanthones from mangosteen as promising cholinesterase inhibitors and their molecular docking studies. *Phytomedicine, 21*(11), 1303-1309. doi:http://dx.doi.org/10.1016/j.phymed.2014.06.017

Klebe, G., & Mietzner, T. (1994). A fast and efficient method to generate biologically relevant conformations. *Journal of Computer-Aided Molecular Design, 8*(5), 583-606. doi:10.1007/BF00123667

Krammer, A., Kirchhoff, P. D., Jiang, X., Venkatachalam, C., & Waldman, M. (2005). LigScore: a novel scoring function for predicting binding affinities. *Journal of Molecular Graphics and Modelling, 23*(5), 395-407.

Lawson, D. (n.d.). *A Brief Introduction to Protein Crystallography by Dave Lawson*. Retrieved from https://www.jic.ac.uk/staff/david-lawson/xtallog/summary.htm

Li, Y., Han, L., Liu, Z., & Wang, R. (2014). Comparative Assessment of Scoring Functions on an Updated Benchmark: 2. Evaluation Methods and General Results. *Journal of Chemical Information and Modeling, 54*, 1717-1736.

Liu, Z., Li, Y., Han, L., L, J., Liu, J., Zhao, Z., . . . Wang, R. (2014). PDB-wide collection of binding data: current status of the PDBbind database. *Bioinformatics, 31*(3), 405-12. doi:10.1093/bioinformatics/btu626

Meng, E. C., Shoichet, B. K., & Kuntz, I. D. (1992). Automated docking with grid-based energy evaluation. *Journal of Computational Chenistry, 13*(4), 505-524. doi:10.1002/jcc.540130412

Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Belew, R. K., & Olson, A. J. (1998). Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry, 19*(4), 1639-1662.

Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R., Goodsell, D. S., & Olson, A. J. (2009). AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. *Journal of Computational Chemistry, 30*(16), 2785-2791. doi:10.1002/jcc.21256

Moustakas, D. T., Lang, P., Pegg, S., Pettersen, E., Kuntz, I. D., Brooijmans, N., & Rizzo, R. C. (2006). Development and validation of a modular, extensible docking program: DOCK 5. *Journal of Computer-Aided Molecular Design, 20*(10), 601-619. doi:10.1007/s10822-006-9060-4

Nakamura, M. T., & Nara, T. Y. (2004). Structure, function, and dietary regulation of Δ6, Δ5, and Δ9 desaturases. *Annual Review Nurtition, 24*, 345-376.

Oganesyan, V., & Zhang, X. (2001, 4 4). *1EUE*. Retrieved from RCSB PDB: http://www.rcsb.org/pdb/explore.do?structureId=1EUE

Pak, Y., & Wang, S. (2000). Application of a molecular dynamics simulation method with a generalized effective potential to the flexible molecular docking problems. *The Journal of Physical Chemistry B, 104*(2), 354-359.

Rarey, M., Kramer, B., Lengauer, T., & Klebe, G. (1996). A fast flexible docking method using an incremental construction algorithm. *Journal of Molecular Biology, 261*(3), 470-489. doi:http://dx.doi.org/10.1006/jmbi.1996.0477

Sousa, S., Ribeiro, A. J., Coimbra, J. T., Neves, R. P., Martins, S., Moorthy, N. H., . . . Ramos, M. J. (2013). Protein-Ligand Docking in the New Millennium – A Retrospective of 10 Years in the Field. *Current Medicinal Chemistry, 20*(18), 2296-2314.

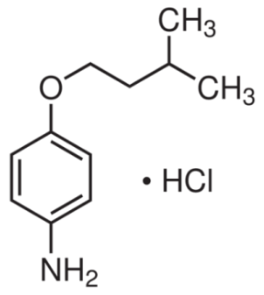
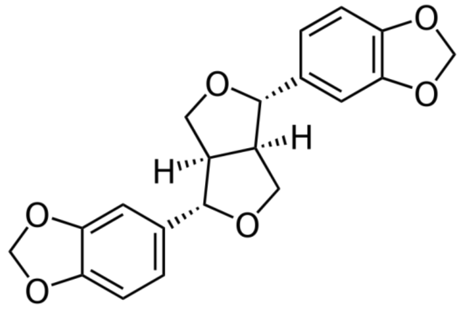
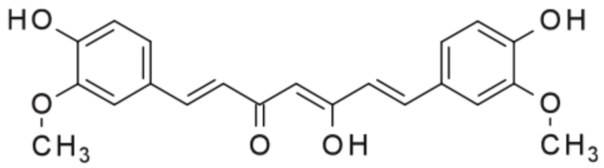
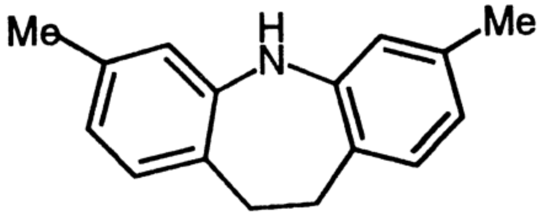
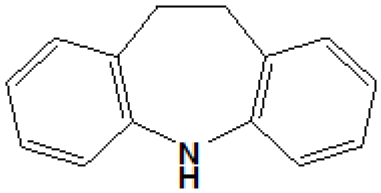
Trosset, J. Y., & Scheraga, H. A. (1999). Prodock: software package for protein modeling and docking. *Journal of Computional Chemistry, 20*, 412-427.

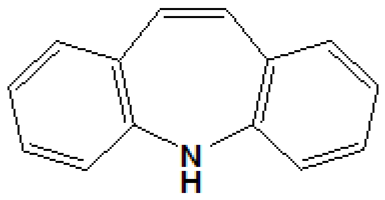
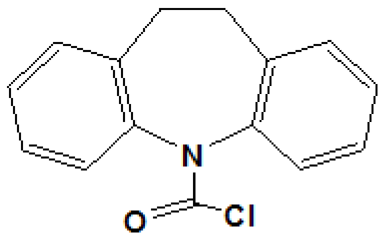
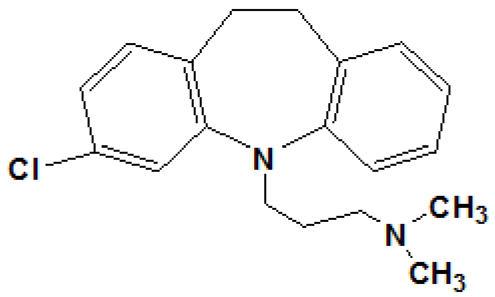
UniProtKB - O60427 (FADS1_HUMAN). (2017, 2 22). Retrieved from UniProt: http://www.uniprot.org/uniprot/O60427

Verdonk, M. L., Cole, J. C., Hartshorn, M. J., Murray, C. W., & Taylor, R. D. (2003). Improved Protein–Ligand Docking Using GOLD. *PROTEINS:Structure, Function, and Genetics*, 609-623.

Zhang, Y. (2008). Progress and challenges in protein structure prediction. *Current Opinion in Structure Biology, 18*(3), 342-348.

APPENDIX A. STRUCTURES OF THE FADS1 LIGANDS

Ligand name	Structure
CP-24879	
Sesamin	
Curcumin	
Dibenzoazepine	
Iminodibenzyl	

Ligand name	Structure
5H-Dibenz[b,f]azepine	
Dibenz[b,f]azepine-5-carbonyl chloride	
clomipramine hydrochloride	

APPENDIX B. PDBIND CORE SET

PDB code	log K_a	protein name
1PS3	2.28	α-mannosidase II
3D4Z	4.89	α-mannosidase II
3EJR	8.57	α-mannosidase II
2QMJ	4.21	maltase-glucoamylase, intestinal
3L4W	6.00	maltase-glucoamylase, intestinal
3L4U	7.52	maltase-glucoamylase, intestinal
3L7B	2.40	glycogen phosphorylase, muscle form
3G2N	4.09	glycogen phosphorylase, muscle form
3EBP	5.91	glycogen phosphorylase, muscle form
2W66	4.05	*O*-glcnacase BT_4395
2WCA	5.60	*O*-glcnacase BT_4395
2VVN	7.30	*O*-glcnacase BT_4395
2X97	5.66	angiotensin converting enzyme
2XHM	6.80	angiotensin converting enzyme
2X8Z	7.96	angiotensin converting enzyme
2X0Y	4.60	*O*-glcnacase NAGJ
2CBJ	8.27	*O*-glcnacase NAGJ
2J62	11.34	*O*-glcnacase NAGJ
3BKK	6.08	angiotensin converting enzyme
3L3N	8.18	angiotensin converting enzyme
2XY9	9.19	angiotensin converting enzyme
1GPK	5.37	acetylcholinesterase
1H23	8.35	acetylcholinesterase
1E66	9.89	acetylcholinesterase
3CJ2	4.85	RNA-dependent RNA polymerase
2D3U	6.92	RNA-dependent RNA polymerase
3GNW	9.10	RNA-dependent RNA polymerase
3F3A	4.19	transporter
3F3C	6.02	transporter
3F3E	7.70	transporter

PDB code	**log K_a**	**protein name**
4GQQ	2.89	α-amylase
1U33	4.60	α-amylase
1XD0	7.12	α-amylase
2WBG	4.45	β-glucosidase A
2J78	6.42	β-glucosidase A
2CET	8.02	β-glucosidase A
2ZXD	5.22	α-l-fucosidase
2ZWZ	7.79	α-l-fucosidase
2ZX6	10.60	α-l-fucosidase
3UDH	2.85	β-secretase 1
4DJV	6.72	β-secretase 1
4GID	10.77	β-secretase 1
3FK1	2.62	3-phosphoshikimate 1-carboxyvinyltransferase
2QFT	5.26	3-phosphoshikimate 1-carboxyvinyltransferase
2PQ9	8.11	3-phosphoshikimate 1-carboxyvinyltransferase
1F8D	3.40	neuraminidase
1F8B	5.40	neuraminidase
1F8C	7.40	neuraminidase
1N2V	4.08	queuine tRNA-ribosyltransferase
1R5Y	6.46	queuine tRNA-ribosyltransferase
3GE7	8.70	queuine tRNA-ribosyltransferase
3HUC	5.99	mitogen-activated protein kinase 14
3GCS	7.25	mitogen-activated protein kinase 14
3E93	8.85	mitogen-activated protein kinase 14
1Q8T	4.76	cAMP-dependent protein kinase
1Q8U	5.96	cAMP-dependent protein kinase
3AG9	8.05	cAMP-dependent protein kinase
3OWJ	6.07	casein kinase II, α subunit
2ZJW	7.70	casein kinase II, α subunit
3PE2	9.76	casein kinase II, α subunit
2V00	3.66	endothiapepsin
3PWW	7.32	endothiapepsin

PDB code	**log *K*a**	**protein name**
3URI	9.00	endothiapepsin
3MFV	2.52	arginase-1
3F80	4.22	arginase-1
3KV2	7.32	arginase-1
2HB1	3.80	protein-tyrosine phosphatase 1b
2QBR	6.33	protein-tyrosine phosphatase 1b
2QBP	8.40	protein-tyrosine phosphatase 1b
3FCQ	2.77	thermolysin
1OS0	6.03	thermolysin
4TMN	10.17	thermolysin
3PXF	4.43	cell division protein kinase 2
2XNB	6.83	cell division protein kinase 2
2FVD	8.52	cell division protein kinase 2
1QI0	2.35	endoglucanase B
1W3K	4.30	endoglucanase 5A
1W3L	6.28	endoglucanase 5A
3IMC	2.96	pantothenate synthetase
3IVG	4.30	pantothenate synthetase
3COY	6.02	pantothenate synthetase
3B3S	2.55	leucyl aminopeptidase
3B3W	4.19	leucyl aminopeptidase
3VH9	6.24	leucyl aminopeptidase
3MSS	4.66	tyrosine-protein kinase ABL1
3K5 V	6.30	tyrosine-protein kinase ABL1
2V7A	8.30	tyrosine-protein kinase ABL1
2BRB	4.86	serine/threonine-protein kinase Chk1
3JVS	6.54	serine/threonine-protein kinase Chk1
1NVQ	8.25	serine/threonine-protein kinase Chk1
3ACW	4.76	dehydrosqualene synthase
2ZCR	6.87	dehydrosqualene synthase
2ZCQ	8.82	dehydrosqualene synthase
1BCU	3.28	thrombin

PDB code	log K_a	protein name
1OYT	7.24	thrombin
3UTU	10.92	thrombin
3U9Q	4.38	peroxisome proliferator-activated receptor γ
2YFE	6.63	peroxisome proliferator-activated receptor γ
2P4Y	9.00	peroxisome proliferator-activated receptor γ
3UO4	6.52	serine/threonine-protein kinase 6
2WTV	8.74	serine/threonine-protein kinase 6
3MYG	10.70	serine/threonine-protein kinase 6
3KGP	2.57	urokinase-type plasminogen activator
1O5B	5.77	urokinase-type plasminogen activator
1SQA	9.21	urokinase-type plasminogen activator
3KWA	4.08	carbonic anhydrase II
2WEG	6.50	carbonic anhydrase II
3DD0	9.00	carbonic anhydrase II
2XDL	3.10	heat shock protein Hsp90-α
1YC1	6.17	heat shock protein Hsp90-α
2YKI	9.46	heat shock protein Hsp90-α
1P1Q	4.89	glutamate receptor 2
3BFU	6.27	glutamate receptor 2
4G8M	7.89	glutamate receptor 2
3G2Z	2.36	β-lactamase
4DE2	4.12	β-lactamase
4DE1	5.96	β-lactamase
1VSO	4.72	glutamate receptor, ionotropic kainate 1
3GBB	6.90	glutamate receptor, ionotropic kainate 1
3FV1	9.30	glutamate receptor, ionotropic kainate 1
2Y5H	5.79	coagulation factor XA
2XBV	8.43	coagulation factor XA
1MQ6	11.15	coagulation factor XA
1LOQ	3.70	orotidine 5′-monophosphate decarboxylase
1LOL	6.39	orotidine 5′-monophosphate decarboxylase
1LOR	11.06	orotidine 5′-monophosphate decarboxylase

PDB code	**log *K*a**	**protein name**
1UTO	2.27	trypsin β
3GY4	5.10	trypsin β
1O3F	7.96	trypsin β
2YGE	5.06	heat shock protein Hsp82
2IWX	6.68	heat shock protein Hsp82
2VW5	8.52	heat shock protein Hsp82
2YMD	3.16	acetylcholine receptor
2XYS	7.42	acetylcholine receptor
2X00	11.33	acetylcholine receptor
2R23	3.72	antibody FAB fragment
3BPC	4.80	antibody FAB fragment
1KEL	7.28	antibody FAB fragment
3OZT	4.13	catechol *O*-methyltransferase
3OE5	6.88	catechol *O*-methyltransferase
3NW9	9.00	catechol *O*-methyltransferase
1ZEA	5.22	antibody FAB fragment
2PCP	8.70	antibody FAB fragment
1IGJ	10.00	antibody FAB fragment
1LBK	3.18	glutathione *S*-transferase P1-1
2GSS	4.94	glutathione *S*-transferase P1-1
10GS	6.40	glutathione *S*-transferase P1-1
3SU5	5.58	NS3/4A protease
3SU2	7.35	NS3/4A protease
3SU3	9.13	NS3/4A protease
3N7A	3.70	3-dehydroquinate dehydratase
3N86	5.64	3-dehydroquinate dehydratase
2XB8	7.59	3-dehydroquinate dehydratase
3AO4	2.07	HIV-1 integrase
3ZSX	3.28	HIV-1 integrase
3ZSO	5.12	HIV-1 integrase
3NQ3	3.78	β-lactoglobulin
3UEU	5.24	β-lactoglobulin

PDB code	**log K_a**	**protein name**
3UEX	6.92	β-lactoglobulin
3LKA	2.82	macrophage metalloelastase (MMP-12)
3EHY	5.85	macrophage metalloelastase (MMP-12)
3F17	8.63	macrophage metalloelastase (MMP-12)
3CFT	4.19	transthyretin
4DES	5.85	transthyretin
4DEW	7.00	transthyretin
3DXG	2.40	ribonuclease A
1W4O	5.22	ribonuclease A
1U1B	7.80	ribonuclease A
3OV1	5.20	growth factor receptor-bound protein 2
3S8O	6.85	growth factor receptor-bound protein 2
1JYQ	8.70	growth factor receptor-bound protein 2
1A30	4.30	HIV-1 protease
3CYX	8.00	HIV-1 protease
4DJR	11.52	HIV-1 protease
3I3B	2.23	β-galactosidase
3MUZ	3.46	β-galactosidase
3VD4	4.82	β-galactosidase
2VO5	4.89	β-mannosidase
2VL4	6.01	β-mannosidase
2VOT	7.14	β-mannosidase
1N1M	5.70	dipeptidyl peptidase 4
2OLE	7.25	dipeptidyl peptidase 4
3NOX	8.66	dipeptidyl peptidase 4
1HNN	6.24	phenylethanolamine *N*-methyltransferase
2G70	7.77	phenylethanolamine *N*-methyltransferase
2OBF	8.85	phenylethanolamine *N*-methyltransferase
1Z95	7.12	androgen receptor
3B68	8.40	androgen receptor
3G0W	9.52	androgen receptor
1SLN	6.64	stromelysin-1

PDB code	**log K_a**	**protein name**
2D1O	7.70	stromelysin-1
1HFS	8.70	stromelysin-1
2JDY	4.37	fucose-binding lectin PA-IIL
2JDM	5.40	fucose-binding lectin PA-IIL
2JDU	6.72	fucose-binding lectin PA-IIL

APPENDIX C. PDBIND CORE SET DOCKING SUMMARY

PDB code	DS Min	DS Num	AutoDock Min	AutoDock Num	Method
10gs	5.8083	0	9.55	0	DS
1a30	9.18772	0	2.86	2	AutoDock
1bcu	6.80629	0	9.56	0	DS
1e66	7.9146	0	NA	0	DS
1f8b	8.6072	0	30.34	0	DS
1f8c	0	10	26.58	0	DS
1f8d	0	1	26.3	0	DS
1gpk	8.21955	0	1.89	10	AutoDock
1h23	6.02768	0	3.33	0	AutoDock
1hfs	11.4727	0	7.28	0	AutoDock
1hnn	8.38057	0	8.03	0	AutoDock
1igj	6.87317	0	21.21	0	DS
1jyq	10.6376	0	12.59	0	DS
1kel	0	14	23.83	0	DS
1lbk	7.91016	0	6.61	0	AutoDock
1lol	5.11139	0	13.26	0	DS
1loq	17.4876	0	17.57	0	DS
1lor	10.7718	0	9.73	0	AutoDock
1mq6	6.54068	0	15.06	0	DS
1n1m	9.32917	0	25.94	0	DS
1n2v	7.73697	0	2.77	2	AutoDock
1nvq	0	5	12.41	0	DS
1o3f	9.99304	0	8.1	0	AutoDock
1o5b	15.5201	0	2.41	1	AutoDock
1os0	12.0796	0	6.8	0	AutoDock
1oyt	11.7536	0	9.7	0	AutoDock
1p1q	7.0728	0	12.3	0	DS
1ps3	NA	0	14.45	0	AutoDock
1q8t	8.47348	0	1.58	4	AutoDock
1q8u	8.55204	0	4.48	0	AutoDock
1qi0	6.72785	0	14.81	0	DS
1r5y	6.00367	0	3.67	0	AutoDock
1sln	9.63172	0	10.67	0	DS
1sqa	8.4153	0	9.8	0	DS
1u1b	8.21593	0	4.4	0	AutoDock
1u33	8.45949	0	9.77	0	DS

PDB code	DS Min	DS Num	AutoDock Min	AutoDock Num	Method
1uto	0	2	1.81	4	DS
1vso	5.22074	0	14.75	0	DS
1w3k	7.56717	0	11.52	0	DS
1w3l	7.89011	0	11.43	0	DS
1w4o	9.42634	0	13.76	0	DS
1xd0	14.0578	0	8.77	0	AutoDock
1yc1	8.0417	0	2.96	1	AutoDock
1z95	12.3223	0	2.42	1	AutoDock
1zea	0	1	27.72	0	DS
2brb	7.57076	0	12.81	0	DS
2cbj	19.6322	0	15.15	0	AutoDock
2cet	7.84718	0	1.07	6	AutoDock
2d1o	9.37148	0	12.66	0	DS
2d3u	0	9	21.57	0	DS
2fvd	7.6841	0	14.1	0	DS
2g70	0	17	7.92	0	DS
2gss	0	16	10.71	0	DS
2hb1	12.5937	0	16.2	0	DS
2iwx	8.08911	0	1.5	10	AutoDock
2j62	9.8234	0	16.73	0	DS
2j78	7.77791	0	0.52	10	AutoDock
2jdm	18.4474	0	22.98	0	DS
2jdu	18.2474	0	22.38	0	DS
2jdy	9.61378	0	25.35	0	DS
2obf	11.2736	0	7.32	0	AutoDock
2ole	9.95456	0	19.51	0	DS
2p4y	12.9	0	18.75	0	DS
2pcp	19.3825	0	22.76	0	DS
2pq9	0	10	1.25	10	DS
2qbp	0	14	12.94	0	DS
2qbr	0	1	14.2	0	DS
2qft	0	20	0.88	8	DS
2qmj	8.65159	0	15.34	0	DS
2r23	10.3994	0	27.12	0	DS
2v00	5.1149	0	0.89	6	AutoDock
2v7a	7.80045	0	9.71	0	DS
2vl4	5.99009	0	11.07	0	DS
2vo5	7.68767	0	13.09	0	DS

PDB code	DS Min	DS Num	AutoDock Min	AutoDock Num	Method
2vot	7.1551	0	15.38	0	DS
2vvn	12.2275	0	12.6	0	DS
2vw5	11.3336	0	3.32	0	AutoDock
2w66	15.9783	0	14.81	0	AutoDock
2wbg	5.95828	0	17.17	0	DS
2wca	6.74984	0	14.59	0	DS
2weg	5.25847	0	0.63	10	AutoDock
2wtv	7.06107	0	8.1	0	DS
2x00	14.4063	0	22.05	0	DS
2x0y	7.80722	0	19.93	0	DS
2x8z	8.5603	0	0.8	9	AutoDock
2x97	11.0496	0	3.69	0	AutoDock
2xb8	7.81645	0	19.86	0	DS
2xbv	13.0473	0	15.71	0	DS
2xdl	7.77039	0	8.66	0	DS
2xhm	NA	0	5.98	0	AutoDock
2xnb	10.4256	0	13.47	0	DS
2xy9	13.6116	0	2.44	3	AutoDock
2xys	9.90856	0	20.24	0	DS
2y5h	7.2362	0	16.04	0	DS
2yfe	8.07678	0	5.7	0	AutoDock
2yge	10.1024	0	3.59	0	AutoDock
2yki	9.01784	0	3.03	0	AutoDock
2ymd	8.78006	0	32.1	0	DS
2zcq	6.39122	0	2.74	1	AutoDock
2zcr	14.4761	0	1.7	5	AutoDock
2zjw	NA	0	14.27	0	AutoDock
2zwz	6.24192	0	29.9	0	DS
2zx6	12.508	0	22.17	0	DS
2zxd	9.0221	0	32.82	0	DS
3acw	12.0784	0	1.36	10	AutoDock
3ag9	12.5201	0	NA	0	DS
3ao4	13.5706	0	14.5	0	DS
3b3s	6.63154	0	13.82	0	DS
3b3w	5.25895	0	NA	0	DS
3b68	0	18	5.25	0	DS
3bfu	7.56331	0	13.7	0	DS
3bkk	11.224	0	2.16	2	AutoDock

PDB code	DS Min	DS Num	AutoDock Min	AutoDock Num	Method
3bpc	10.5394	0	27.49	0	DS
3cft	17.2524	0	9.54	0	AutoDock
3cj2	11.1617	0	23.13	0	DS
3coy	8.15473	0	15.87	0	DS
3cyx	13.1025	0	3.41	0	AutoDock
3d4z	25.8236	0	15.46	0	AutoDock
3dd0	7.7338	0	1.97	5	AutoDock
3dxg	7.25655	0	2.4	7	AutoDock
3e93	12.3006	0	6.8	0	AutoDock
3ebp	12.1055	0	28.36	0	DS
3ehy	10.1327	0	10.88	0	DS
3ejr	6.38707	0	14.8	0	DS
3f17	11.5185	0	11.99	0	DS
3f3a	0	20	23.15	0	DS
3f3c	9.28237	0	22.52	0	DS
3f3e	NA	0	21.59	0	AutoDock
3f80	0	52	18.96	0	DS
3fcq	7.52613	0	3.62	0	AutoDock
3fk1	5.29645	0	0.4	10	AutoDock
3fv1	9.93798	0	0.32	10	AutoDock
3g0w	12.7258	0	4.47	0	AutoDock
3g2n	23.9442	0	1.73	10	AutoDock
3g2z	4.63534	0	15.38	0	DS
3gbb	10.5483	0	0.43	10	AutoDock
3gcs	10.8583	0	5.66	0	AutoDock
3ge7	7.0495	0	23.02	0	DS
3gnw	9.27252	0	1.91	4	AutoDock
3gy4	16.8961	0	8.81	0	AutoDock
3huc	9.44927	0	10.58	0	DS
3i3b	6.05309	0	NA	0	DS
3imc	6.26741	0	19.98	0	DS
3ivg	22.181	0	15.44	0	AutoDock
3jvs	0	2	13.1	0	DS
3k5v	8.92985	0	19.47	0	DS
3kgp	14.5175	0	7.83	0	AutoDock
3kv2	12.6006	0	22.8	0	DS
3kwa	11.6569	0	1.92	8	AutoDock
3l3n	12.2073	0	2.43	3	AutoDock

PDB code	DS Min	DS Num	AutoDock Min	AutoDock Num	Method
3l4u	NA	0	14.56	0	AutoDock
3l4w	14.2631	0	17.88	0	DS
3l7b	NA	0	27.15	0	AutoDock
3lka	7.87774	0	3.14	0	AutoDock
3mfv	14.4811	0	19.31	0	DS
3mss	27.0544	0	25.81	0	AutoDock
3muz	16.7868	0	NA	0	DS
3myg	7.28649	0	9.15	0	DS
3n7a	7.08678	0	17	0	DS
3n86	15.6088	0	21.61	0	DS
3nox	19.8031	0	20.25	0	DS
3nq3	8.36094	0	1.38	6	AutoDock
3nw9	12.6828	0	6.64	0	AutoDock
3oe5	0	34	5.35	0	DS
3ov1	6.37898	0	8.3	0	DS
3owj	10.8669	0	14.54	0	DS
3ozt	0	40	3.17	0	DS
3pe2	9.7509	0	13.94	0	DS
3pww	9.44403	0	4.3	0	AutoDock
3pxf	8.98203	0	20.31	0	DS
3s8o	NA	0	8.49	0	AutoDock
3su2	13.0231	0	7.91	0	AutoDock
3su3	17.0831	0	9.19	0	AutoDock
3su5	16.5936	0	9.98	0	AutoDock
3u9q	8.08309	0	11.47	0	DS
3udh	5.78807	0	1.38	10	AutoDock
3ueu	20.6858	0	19.84	0	AutoDock
3uex	22.0354	0	20.56	0	AutoDock
3uo4	8.22252	0	6.48	0	AutoDock
3uri	NA	0	6.76	0	AutoDock
3utu	16.3407	0	11.15	0	AutoDock
3vd4	10.8686	0	NA	0	DS
3vh9	5.65174	0	12.58	0	DS
3zso	0	24	14.01	0	DS
3zsx	15.2521	0	14.14	0	AutoDock
4de1	5.94607	0	13.59	0	DS
4de2	7.02555	0	14	0	DS
4des	18.8617	0	7.27	0	AutoDock

PDB code	DS Min	DS Num	AutoDock Min	AutoDock Num	Method
4dew	11.5001	0	6.62	0	AutoDock
4djr	11.207	0	3.2	0	AutoDock
4djv	7.47272	0	3.3	0	AutoDock
4g8m	7.71106	0	0.51	10	AutoDock
4gid	16.5132	0	3.63	0	AutoDock
4gqq	17.2052	0	18.23	0	DS
4tmn	9.39099	0	6.34	0	AutoDock

APPENDIX D. PYTHON CODE FOR RECEPTOR PREPARATION IN AUTODOCK

```
# prepare_receptor4.py
import os

from MolKit import Read
import MolKit.molecule
import MolKit.protein
from AutoDockTools.MoleculePreparation import AD4ReceptorPreparation


if __name__ == '__main__':
    import sys
    import getopt



    def usage():
        "Print helpful, accurate usage statement to stdout."
        print "Usage: prepare_receptor4.py -r filename"
        print
        print "    Description of command..."
        print "         -r    receptor_filename "
        print "         supported file types include pdb,mol2,pdbq,pdbqs,pdbqt, possibly
pqr,cif"
        print "    Optional parameters:"
        print "        [-v]   verbose output (default is minimal output)"
        print "        [-o pdbqt_filename]   (default is 'molecule_name.pdbqt')"
        print "        [-A]   type(s) of repairs to make: "
        print "               'bonds_hydrogens': build bonds and add hydrogens "
        print "               'bonds': build a single bond from each atom with no bonds to its
closest neighbor"
        print "               'hydrogens': add hydrogens"
        print "               'checkhydrogens': add hydrogens only if there are none already"
        print "               'None': do not make any repairs "
        print "               (default is 'None')"
        print "        [-C]   preserve all input charges ie do not add new charges "
        print "               (default is addition of gasteiger charges)"
        print "        [-p]   preserve input charges on specific atom types, eg -p Zn -p Fe"
        print "        [-U]   cleanup type:"
        print "                    'nphs': merge charges and remove non-polar hydrogens"
        print "                    'lps': merge charges and remove lone pairs"
        print "                    'waters': remove water residues"
        print "                    'nonstdres': remove chains composed entirely of residues of"
        print "                              types other than the standard 20 amino acids"
        print "                       'deleteAltB': remove XX@B atoms and rename XX@A
atoms->XX"
        print "                    (default is 'nphs_lps_waters_nonstdres') "
        print "               [-e]   delete every nonstd residue from any chain"
        print "                      'True': any residue whose name is not in this list:"
        print "                              ['CYS','ILE','SER','VAL','GLN','LYS','ASN', "
        print "                              'PRO','THR','PHE','ALA','HIS','GLY','ASP', "
        print "                              'LEU', 'ARG', 'TRP', 'GLU', 'TYR','MET', "
        print "                              'HID', 'HSP', 'HIE', 'HIP', 'CYX', 'CSS']"
        print "                      will be deleted from any chain. "
        print "                      NB: there are no    nucleic acid residue names at all "
        print "                      in the list and no metals. "
        print "                    (default is False which means not to do this)"
        print "               [-M]   interactive "
        print "                    (default is 'automatic': outputfile is written with no further user
input)"
        print "                    [-d dictionary_filename] file to contain receptor summary
information"


    # process command arguments
    try:
        opt_list, args = getopt.getopt(sys.argv[1:], 'r:vo:A:Cp:U:eM:d:')

    except getopt.GetoptError, msg:
        print 'prepare_receptor4.py: %s' %msg
        usage()
        sys.exit(2)

    files = os.listdir('C:\Users\wang28\Desktop\left')
#      mol = []
    for file in files:
        # ligand_filename =    None
        receptor_filename = os.path.join("C:\\Users\\wang28\\Desktop\\left\\", file)# initialize
required parameters
        #-s: receptor
        #receptor_filename =    None

        # optional parameters
        verbose = None
        #-A: repairs to make: add bonds and/or hydrogens or checkhydrogens
        repairs = ''
        #-C default: add gasteiger charges
        charges_to_add = 'gasteiger'
        #-p preserve charges on specific atom types
        preserve_charge_types=None
        #-U: cleanup by merging nphs_lps, nphs, lps, waters, nonstdres
        cleanup    = "nphs_lps_waters_nonstdres"
        #-o outputfilename
        outputfilename = None
        #-m mode
        mode = 'automatic'
        #-e delete every nonstd residue from each chain
        delete_single_nonstd_residues = None
        #-d dictionary
        dictionary = None

        #'r:vo:A:Cp:U:eMh'
        for o, a in opt_list:
            if o in ('-r', '--r'):
                receptor_filename = a
                if verbose: print 'set receptor_filename to ', a
            if o in ('-v', '--v'):
                verbose = True
                if verbose: print 'set verbose to ', True
            if o in ('-o', '--o'):
                outputfilename = a
                if verbose: print 'set outputfilename to ', a
            if o in ('-A', '--A'):
                repairs = a
                if verbose: print 'set repairs to ', a
            if o in ('-C', '--C'):
                charges_to_add = None
                if verbose: print 'do not add charges'
        if o in ('-p', '--p'):
                if not preserve_charge_types:
                        preserve_charge_types = a
                else:
                        preserve_charge_types = preserve_charge_types + ','+ a
                if verbose: print 'preserve initial charges on ', preserve_charge_types
        if o in ('-U', '--U'):
                cleanup    = a
                if verbose: print 'set cleanup to ', a
        if o in ('-e', '--e'):
                delete_single_nonstd_residues    = True
                if verbose: print 'set delete_single_nonstd_residues to True'
        if o in ('-M', '--M'):
                mode = a
                if verbose: print 'set mode to ', a
        if o in ('-d', '--d'):
                dictionary    = a
                if verbose: print 'set dictionary to ', dictionary
        if o in ('-h', '--'):
                usage()
                sys.exit()


if not receptor_filename:
        print 'prepare_receptor4: receptor filename must be specified.'
        usage()
        sys.exit()


mols = Read(receptor_filename)
if verbose: print 'read ', receptor_filename
mol = mols[0]
preserved = {}
if charges_to_add is not None and preserve_charge_types is not None:
        preserved_types = preserve_charge_types.split(',')
        if verbose: print "preserved_types=", preserved_types
        for t in preserved_types:
                if verbose: print 'preserving charges on type->', t
                if not len(t): continue
                ats = mol.allAtoms.get(lambda x: x.autodock_element==t)
                if verbose: print "preserving charges on ", ats.name
                for a in ats:
                    if a.chargeSet is not None:
                        preserved[a] = [a.chargeSet, a.charge]

        if len(mols)>1:
            if verbose: print "more than one molecule in file"
            #use the molecule with the most atoms
            ctr = 1
            for m in mols[1:]:
                ctr += 1
                if len(m.allAtoms)>len(mol.allAtoms):
                    mol = m
                    if verbose: print "mol set to ", ctr, "th molecule with",
len(mol.allAtoms), "atoms"
        mol.buildBondsByDistance()

        if verbose:
            print "setting up RPO with mode=", mode,
            print "and outputfilename= ", outputfilename
            print "charges_to_add=", charges_to_add
            print "delete_single_nonstd_residues=", delete_single_nonstd_residues

        RPO = AD4ReceptorPreparation(mol, mode, repairs, charges_to_add,
                                    cleanup, outputfilename=outputfilename,
                                    preserved=preserved,

delete_single_nonstd_residues=delete_single_nonstd_residues,
                                    dict=dictionary)

        if charges_to_add is not None:
            #restore any previous charges
            for atom, chargeList in preserved.items():
                atom._charges[chargeList[0]] = chargeList[1]
                atom.chargeSet = chargeList[0]
```

```
# To execute this command type:
# prepare_receptor4.py -r pdb_file -o outputfilename -A checkhydrogens
```

APPENDIX E. PYTHON CODE FOR LIGAND PREPARATION IN AUTODOCK

```
# prepare_ligand4.py
import os

from MolKit import Read

from AutoDockTools.MoleculePreparation import AD4LigandPreparation



if __name__ == '__main__':
        import sys
        import getopt


        def usage():
                "Print helpful, accurate usage statement to stdout."
                print "Usage: prepare_ligand4.py -l filename"
                print
                print "     Description of command..."
                print "           -l        ligand_filename (.pdb or .mol2 or .pdbq format)"
                print "     Optional parameters:"
                print "          [-v]      verbose output"
                print "          [-o pdbqt_filename] (default output filename is ligand_filename_stem
+ .pdbqt)"
                print "          [-d]      dictionary to write types list and number of active torsions "

                print "          [-A]      type(s) of repairs to make:\n\t\t bonds_hydrogens, bonds,
hydrogens (default is to do no repairs)"
                print "          [-C]      do not add charges (default is to add gasteiger charges)"
                print "          [-p]      preserve input charges on atom type, eg -p Zn"
                print "                    (default is not to preserve charges on any specific atom type)"
                print "          [-U]      cleanup type:\n\t\t nphs_lps, nphs, lps, " (default is 'nphs_lps')
"
                print "          [-B]      type(s) of bonds to allow to rotate "
                print "                    (default sets 'backbone' rotatable and 'amide' + 'guanidinium'
non-rotatable)"
                print "          [-R]      index for root"
                print "              [-F]       check for and use largest non-bonded fragment (default is not
to do this)"
                print "              [-M]        interactive (default is automatic output)"
                print "              [-I]       string of bonds to inactivate composed of "
                print "                                 of zero-based atom indices eg 5_13_2_10     "
                print "                                 will inactivate atoms[5]-atoms[13] bond "
                print "                                                  and atoms[2]-atoms[10] bond "
                print "                                      (default is not to inactivate any specific bonds)"
                print "              [-Z]       inactivate all active torsions          "
                print "                                      (default is leave all rotatable active except amide and
guanidinium)"
                print "              [-g]       attach all nonbonded fragments "
                print "              [-s]       attach all nonbonded singletons: "
                print "                                 NB: sets attach all nonbonded fragments too"
                print "                                      (default is not to do this)"


        # process command arguments
        try:
                opt_list, args = getopt.getopt(sys.argv[1:], 'l:vo:d:A:Cp:U:B:R:MFI:Zgsh')
        except getopt.GetoptError, msg:
                print 'prepare_ligand4.py: %s' %msg
                usage()
                sys.exit(2)

        # initialize required parameters
        #-l: ligand



        files = os.listdir('C:\Users\wang28\Desktop\PDbind\ligand')
        mol = []
        for file in files:
                # ligand_filename =    None
                ligand_filename = os.path.join("C:\\Users\\wang28\\Desktop\\PDbind\\ligand\\", file)

                # optional parameters
                verbose = None
                add_bonds = False
                #-A: repairs to make: add bonds and/or hydrogens
```

```
repairs = ""
#-C    default: add gasteiger charges
charges_to_add = 'gasteiger'
#-p preserve charges on specific atom types
preserve_charge_types=''
#-U: cleanup by merging nphs_lps, nphs, lps
cleanup    = "nphs_lps"
#-B named rotatable bond type(s) to allow to rotate
#allowed_bonds = ""
allowed_bonds = "backbone"
#-r    root
root = 'auto'
#-o outputfilename
outputfilename = None
#-F check_for_fragments
check_for_fragments = False
#-I bonds_to_inactivate
bonds_to_inactivate = ""
#-Z inactivate_all_torsions
inactivate_all_torsions = False
#-g attach_nonbonded_fragments
attach_nonbonded_fragments = False
#-s attach_nonbonded_singletons
attach_singletons = False
#-m mode
mode = 'automatic'
#-d dictionary
dict = None

#'l:vo:d:A:CKU:B:R:MFI:Zgs'
for o, a in opt_list:
        #print "o=", o, " a=", a
        if o in ('-l', '--l'):
                ligand_filename = a
                if verbose: print 'set ligand_filename to ', a
        if o in ('-v', '--v'):
                verbose = True
                if verbose: print 'set verbose to ', True
        if o in ('-o', '--o'):
```

```
        outputfilename = a
        if verbose: print 'set outputfilename to ', a
    if o in ('-d', '--d'):
        dict = a
        if verbose: print 'set dict to ', a
    if o in ('-A', '--A'):
        repairs = a
        if verbose: print 'set repairs to ', a
    if o in ('-C', '--C'):
        charges_to_add = None
        if verbose: print 'do not add charges'
    if o in ('-p', '--p'):
        preserve_charge_types+=a
        preserve_charge_types+=','
        if verbose: print 'preserve initial charges on ', preserve_charge_types
    if o in ('-U', '--U'):
        cleanup   = a
        if verbose: print 'set cleanup to merge ', a
    if o in ('-B', '--B'):
        allowed_bonds = a
        if verbose: print 'allow ', a, 'bonds set to rotate'
    if o in ('-R', '--R'):
        root = a
        if verbose: print 'set root to ', root
    if o in ('-F', '--F'):
        check_for_fragments = True
        if verbose: print 'set check_for_fragments to True'
    if o in ('-M', '--M'):
        mode = a
        if verbose: print 'set mode to ', a
    if o in ('-I', '--I'):
        bonds_to_inactivate = a
        if verbose: print 'set bonds_to_inactivate to ', a
    if o in ('-Z', '--Z'):
        inactivate_all_torsions = True
        if verbose: print 'set inactivate_all_torsions to ', inactivate_all_torsions
    if o in ('-g', '--g'):
        attach_nonbonded_fragments = True
```

```
            if verbose: print 'set attach_nonbonded_fragments to ',
attach_nonbonded_fragments
        if o in ('-s', '--s'):
            attach_singletons = True
            if verbose: print 'set attach_singletons to ', attach_singletons
        if o in ('-h', '--'):
            usage()
            sys.exit()


    if not  ligand_filename:
        print 'prepare_ligand4: ligand filename must be specified.'
        usage()
        sys.exit()

    if attach_singletons:
        attach_nonbonded_fragments = True
        if verbose: print "using attach_singletons so attach_nonbonded_fragments also"

    mols = Read(ligand_filename)
    if verbose: print 'read ', ligand_filename
    mol = mols[0]
    if len(mols)>1:
        if verbose:
            print "more than one molecule in file"
        #use the one molecule with the most atoms
        ctr = 1
        for m in mols[1:]:
            ctr += 1
            if len(m.allAtoms)>len(mol.allAtoms):
                mol = m
                if verbose:
                    print "mol set to ", ctr, "th molecule with", len(mol.allAtoms),
"atoms"
    coord_dict = {}
    for a in mol.allAtoms: coord_dict[a] = a.coords


    mol.buildBondsByDistance()
```

```
        if charges_to_add is not None:
            preserved = {}
            preserved_types = preserve_charge_types.split(',')
            for t in preserved_types:
                if not len(t): continue
                ats = mol.allAtoms.get(lambda x: x.autodock_element==t)
                for a in ats:
                    if a.chargeSet is not None:
                        preserved[a] = [a.chargeSet, a.charge]



        if verbose:
            print "setting up LPO with mode=", mode,
            print "and outputfilename= ", outputfilename
            print "and check_for_fragments=", check_for_fragments
            print "and bonds_to_inactivate=", bonds_to_inactivate
        LPO = AD4LigandPreparation(mol, mode, repairs, charges_to_add,
                                    cleanup, allowed_bonds, root,
                                    outputfilename=outputfilename,
                                    dict=dict, check_for_fragments=check_for_fragments,
                                    bonds_to_inactivate=bonds_to_inactivate,
                                    inactivate_all_torsions=inactivate_all_torsions,

attach_nonbonded_fragments=attach_nonbonded_fragments,
                                    attach_singletons=attach_singletons)
        #do something about atoms with too many bonds (?)
        #FIX THIS: could be peptide ligand (???)
        #          ??use isPeptide to decide chargeSet??
        if charges_to_add is not None:
            #restore any previous charges
            for atom, chargeList in preserved.items():
                atom._charges[chargeList[0]] = chargeList[1]
                atom.chargeSet = chargeList[0]
        if verbose: print "returning ", mol.returnCode
        bad_list = []
        for a in mol.allAtoms:
            if a in coord_dict.keys() and a.coords!=coord_dict[a]:
                bad_list.append(a)
        if len(bad_list):
            print len(bad_list), ' atom coordinates changed!'
            for a in bad_list:
                print a.name, ":", coord_dict[a], ' -> ', a.coords
        else:
            if verbose: print "No change in atomic coordinates"
        if mol.returnCode!=0:
            sys.stderr.write(mol.returnMsg+"\n")
    sys.exit(mol.returnCode)


# To execute this command type:
# prepare_ligand4.py -l pdb_file -v
```