In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

Computer Aided Drug Discovery and Molecular Simulation:

Software Development and Applications

by

Aoxiang Tao

A dissertation

submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy in the Department of Biomedical and Pharmaceutical Sciences

Idaho State University

Fall 2020

Copyright (2020) Aoxiang Tao

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Aoxiang Tao find it satisfactory and recommend that it be accepted.

_____

Dong Xu, Ph.D.
Major Advisor

_____

Vaughn L. Culbertson, Pharm.D.
Committee Member

_____

Robin Dodson, Ph.D.
Committee Member

_____

Srinath Pashikanti, Ph.D.
Committee Member

_____

Michael J. Aldape, Ph.D.
Graduate Faculty Representative

Dedication

This dissertation is dedicated to my family. I love you-all to infinity and beyond.

Acknowledgments

I would first like to acknowledge my advisor Dr. Dong Xu. As the ancient Chinese saying goes, “once master, father for life”, Dr. Xu has been fathering me since I came to Idaho State University.

I don’t know if true random exists in this universe, just like we don’t know if God can roll the dice. God is always in the believer’s heart, but dice are not always in our hands because gambling is not good for most of us. At the crossroad of my desperate life in 2015, Dr. Xu gave me the dice and I rolled them. Without Dr. Xu’s help, my life would be completely different, and none of the amazing things would happen. He has set goals and solved many troubles for my research projects, to help me reach my potential. With his guidance, I have been learning many techniques in the field of computational chemistry and we have published many of our projects in the last couple of years. Man plans, God laughs. As a part-time atheist, I do not really have a plan for myself. Fortunately, Dr. Xu always helps me make plans and turn those plans into actions. Without Dr. Xu, my name would not be in any of the publications. Not only does he guide me at work and school, but he also helps me a lot in life. Here, on behalf of all my family members, I want to thank Dr. Xu, for giving all of us the precious opportunity. Without Dr. Xu, some of my kids would not even exist in this universe.

In the Xu Laboratory, Matthew Caylor has been providing so much help as a technician. I would like to thank Matthew Caylor, for providing all of those help in the laboratory. Caylor has been fixing most of the hardware issues, setting up software and work environment, and helping programming in many of my projects. Caylor has always

found time to help me and give me his own opinion in discussions. Without the help from Caylor, I would have to spend more time to solve problems in research projects.

My fellow graduate students in the Xu Laboratory have been very helpful in my research projects. Yuying Huang and Shaikh Emdadur Rahman have been providing a lot of intellectual and emotional support since the year 2017. I would like to thank you guys for being awesome teammates.

Former Xu Lab members Peter Economen, Hung Dang, Roy Johnson, Alex Ham, and Patrick Erstad helped me a lot in life when I came to Boise. Former Xu Lab Yasuhiro Shinohara provided a lot of help in the ezCADD project.

I would also like to thank my committee members Dr. Robin Dodson, Dr. Vaughn Culbertson, Dr. Srinath Pashikanti, and Dr. Michael Aldape. Thank you all for your time and support. Dr. Robin Dodson was always very energetic that he walked over the speed limit in the building. Dr. Vaughn Culbertson looked like Nicolas Cage to me, and he left many unopened TV dinners in the freezer. Dr. Srinath Pashikanti was very kind, and we had good discussions about the design of assignments for the PharmD course. Dr. Michael Aldape was a funny but humble person, and he played many funny jokes.

At last, I want to thank all of you again. It was an amazing journey in Idaho.

Table of Contents

List of Figures

List of Tables

List of Equations

Computer Aided Drug Discovery and Molecular Simulation:

Software Development and Applications

Dissertation Abstract -- Idaho State University (2020)

Computer-aided drug design (CADD) is an indispensable part of drug discovery, which provides advanced techniques, for example, molecular modeling and high throughput virtual screening. Applying CADD techniques can significantly reduce the time and cost in the study of drug design, and impressively improve the success rate in the research and development of the drug. However, many experimental researchers in the research area of drug discovery have limited experience of CADD applications and very basic understandings of CADD, due to the expenses of CADD software packages and hardware, the complexity of CADD techniques, and the difficulty in learning and training for CADD techniques.

To eliminate the technical and fiscal barriers for underserved researchers who want to apply CADD to their studies, we have developed a web-based CADD environment (ezCADD) which provides graphical CADD applications. ezCADD simplifies the complexity of CADD, which enables researchers to perform CADD tasks without the distress of the requirements of (1) computational background; (2) computer hardware and software purchase; (3) software compilation, installation, and update; and (4) computer system maintenance. To date, we have developed several fundamental CADD applications for the ezCADD platform, for example, small-molecule docking (ezSMDock), binding site detection (ezPocket), protein-protein docking (ezPPDock), high throughput virtual screening (ezHTVS), drug target and poly-pharmacology identification (ezPocketSearch) and other applications. Those applications have drawn

researchers' attention from all over the world. More than 20,000 CADD jobs have been executed on the ezCADD platform by more than 2000 users from different countries.

A variety of CADD techniques have been applied in TMC1 and Mcs1 studies, along with experimental studies from collaborators. In the TMC1 study, we built a homology model of zebrafish TMC1 protein. Then molecular dynamics simulations and high throughput virtual screening were performed with the homology model, and machine learning and QSAR methods were also applied with the existing compound data. In the Mcs1 study, a high-quality Mcs1 protein model and a full-length VCAM-1 model were built. Molecular dynamics simulations were performed with the homology models. Then the very plausible conformations of the Mcs1 and VCAM-1 were extracted from simulation trajectories for docking studies.

Key Words: Computer Aided Drug Discovery, Drug Design, Molecular Dynamics, Homology Modeling, Machine Learning, High Throughput Virtual Screening.

Chapter I: ezCADD Development

I.1 Introduction

Computer-Aided Drug Design

Computer-aided drug design (CADD) has become a powerful tool in the realm of modern drug discovery and rational drug design, along with the development of computer sciences and the increase of computing power. With the facilitation of computer modeling techniques and *in silico* studies to drug discovery, the importance of understanding the mechanism of drug-receptor interactions has been gradually increasing. Using advanced CADD techniques can dramatically increase the hit rate, decrease the cost and human efforts in high throughput drug screening, comparing to a large number of workloads for conventional benchtop experiments in the wet lab (Cheng, Li, Zhou, Wang, & Bryant, 2012). Combining with explosive increments of data, machine learning techniques have been applied in many scientific research areas in recent decades (Lo, Rensi, Torng, & Altman, 2018). In drug discovery, machine learning has become a powerful tool for researchers.

Based on the knowledge of the biological target receptor and the ligand molecule that binds to the target receptor, CADD can be mainly divided into two categories, structure-based drug design (SBDD) and ligand-based drug design (LBDD) (Kapetanovic, 2008; Macalino, Gosu, Hong, & Choi, 2015; Marshall, 1987; Merz Jr, Ringe, & Reynolds, 2010; Veselovsky & Ivanov, 2003).

Structure-Based Drug Design

Structure-based drug design, also known as direct drug design, is based on the three dimensional (3D) structure of the target receptor which can be not only experimentally determined by X-ray crystallography, nuclear magnetic resonance (NMR) spectroscopy, and electron microscopy (EM), but also computationally predicted by *ab initio* protein modeling and homology modeling (also known as comparative protein modeling). A potential ligand binding site of target protein will be identified with many characters, such as the shape and size of pocket or cavity, amount and location of hydrogen bond donors and acceptors, and hydrophobicity of the binding site. The ligand binding site can be either an active site (also known as the orthosteric site) or allosteric site, and the ligand can be designed as an agonist or antagonist depending on the therapeutic effects. In structure-based drug design, a molecular docking study or high throughput virtual screening (HTVS) task usually will be performed with a library of a large number of compounds to evaluate the binding affinity of ligand and investigate the relationship between receptor and ligand (Anderson, 2003; Konc & Janežič, 2014; Merz Jr et al., 2010).

Ligand-Based Drug Design

Ligand-based drug design, also known as indirect drug design, is based on the information of known ligand molecules without the necessity of understanding the target receptor or drug-receptor relationship. The 3D structure of the target receptor can be very difficult to obtain due to a variety of reasons, such as the large size of protein, difficulties in crystallization, or other technical difficulties (Macalino et al., 2015; Merz Jr et al., 2010). Therefore, ligand-based drug design is the only and best option for many research

studies. Besides, ligand-based drug design can be applied in the study where multiple different target receptors have been involved, which can be extremely difficult in the structure-based drug design study. Quantitative structure-activity relationship (QSAR), pharmacophore modeling, and ligand-based virtual screening (LBVS) are the widely used ligand-based drug design techniques (Macalino et al., 2015).

A statistical QSAR model can be generated based on the correlation between bioactivity data, such as absorption, distribution, metabolism, and excretion (ADME) properties, and structural variation of a set of compounds. Although a QSAR model does not need a target receptor information, some molecular data are still required to generate a good QSAR model, such as the amount of compounds with activity data from experiments, molecular descriptors for the compounds, and selection of training and test datasets. Similar to QSAR modeling, pharmacophore modeling can screen similar compounds in 3D arrangement with different scaffolds. Based on the 3D alignment of ligands, key features and mechanistic properties of pharmacophore can be revealed (Macalino et al., 2015; Veselovsky & Ivanov, 2003).

Ligand-based virtual screening also utilizes structural similarity of the ligand molecules. Two-dimensional (2D) or 3D molecular fingerprints usually will be used in similarity search against a large database to find compounds that structurally similar to the query molecule. Molecular structural similarity search is extremely fast and inexpensive when using high-performance graphics processing unit (GPU) (Lavecchia, 2015). A molecular structural similarity search against a database with millions of compounds can be finished in minutes.

Machine Learning

Machine learning, a new and hot topic in the realm of artificial intelligence, has provided a variety of tools that facilitate the development of drug discovery (Vamathevan et al., 2019). A variety of machine learning techniques (such as random forest, decision tree, support vector machine, relevance vector machine, neural network, etc.) have been well applied and developed in the applications for QSAR modeling, and quantitative structure-property relationship (QSPR) modeling (Agarwal, Dugar, & Sengupta, 2010). Although sometimes controversial, machine learning techniques have been widely applied in docking-based virtual screening (Cheng et al., 2012).

As an ensemble machine learning technique, the random forest method constructs many decision trees to classify the entire dataset (or learning sample) with the majority vote (details as illustrated in Figure 1). Each of the decision tree nodes in the random forest is independent of other nodes. A subset of attributes will be randomly selected by each node from the entire set of attributes (Oshiro, Perez, & Baranauskas, 2012; Svetnik et al., 2003). Unlike its predecessor, decision tree learning, random forest averages multiple independent decision trees to reduce the high variance and irregular patterns from each decision tree. The random forest method can be used in both classification and regression models (Svetnik, Liaw, Tong, & Wang, 2004).

Similar to random forest, support vector machine (SVM) is another supervised machine learning method for both classification and regression models. SVM is fairly straightforward to apply to many scientific studies with high performance (Matsumoto, Aoki, & Ohwada, 2016). Researchers in life science have been using the SVM method for classifying objects such as protein and DNA sequences, mass spectrums, and micro-

array expression profiles (Noble, 2006). In drug discovery, the SVM method can be used for the computational identification of active compounds and the prediction of chemical and biological properties (Heikamp & Bajorath, 2014; Matsumoto et al., 2016).

Artificial Neural Network

Artificial neural network (ANN), as a novel machine learning model, is designed to mimic the biological neural networks that constitute the human brain to process information (schematic of ANN is illustrated in Figure 2) (Terfloth & Gasteiger, 2001). The basic unit of the neural network is called the artificial neuron, also known as a node, which mimics a neuron in the human brain. The connection between neurons called an edge, which is similar to the biological synapse, transmits signals from the presynaptic neuron to the postsynaptic neuron. The artificial neuron can receive a signal, process it, and then pass it to the next connected neuron. The artificial neural networks consist of three different layers: one input layer, one output layer, and one hidden layer that processing the signals from the input layer and passing the result to the output layer (Hessler & Baringhaus, 2018).

Deep neural network (DNN) is based on artificial neural networks, but with multiple hidden layers between the input layer and the output layer. Recurrent neural network (RNN) is another kind of artificial neural networks, where the connections between nodes are directed. In RNN, signals can travel in loops from layer to layer, which is different from any of feedforward neural networks, where signals can only travel from layer to the next layer in one direction (Hessler & Baringhaus, 2018; Manallack & Livingstone, 1999; Terfloth & Gasteiger, 2001). Deep learning techniques are widely used in many research areas and our daily life, such as speech recognition, facial

recognition, natural language processing, automatic translation, gaming, and so on. In drug discovery, deep learning techniques are also applied in many research studies, such as property prediction, QSAR and QSPR modeling, and *de novo* drug design (Hessler & Baringhaus, 2018). Although widely applied in many fields, there are still many challenges for ANN. Overfitting and computing time are the two very common issues in ANN modeling studies, including DNN and RNN (Manallack & Livingstone, 1999).

ezCADD

CADD has been applied in many different research fields of drug discovery for decades, with its powerful computation capacity. However, the majority of experimental biomedical researchers, including but not limited to medicinal chemists, biochemists, cell biologists, pharmacologists, and toxicologists, have little to no computational background. Many researchers, particularly in academia, may not have access to a computational collaborator or team member. Although CADD software is relatively abundant, the more user-friendly packages are likely to be commercial ones, costing from hundreds to thousands of US dollars per year at academic pricing. On the other hand, most free or open-source CADD software is designed for Unix/Linux operating system, requiring users' technical ability to maintain a Unix/Linux system, compile, and install the software directly from source code. Furthermore, regardless of the choice of commercial or free software, a biomedical researcher would still need to purchase a powerful computer, maintain the computer system, and perform software updates. Therefore, access to CADD depends on (1) technical expertise and (2) funding availability for the acquisition of a good computer system and/or commercial software (Tao et al., 2019).

Due to these technical and fiscal barriers, it is clear that a large population of biomedical researchers around the world, particularly those from developing countries, are underserved. Without access to CADD, their drug discovery work may be hindered or less efficient. To address this research disparity and the need to help biomedical researchers overcome these barriers, we have developed a web-based CADD environment (ezCADD) with the goal to implement and provide a suite of CADD applications essential to drug discovery. The choice of web service as the ezCADD platform has many advantages over other types of application delivery to the end-users. ezCADD enables access to CADD by removing the traditional requirements of (1) computational background; (2) computer purchase; (3) software purchase; (4) software compilation or installation; (5) software update; and (6) computer system maintenance (Tao et al., 2019).

Upon searching for recent publications and a directory of open-source molecular modeling software (Pirhadi, Sunseri, & Koes, 2016), there are only a small number of 3D visualization-enabled web services similar to ezCADD. PlayMolecule.org offers multiple web applications for protein preparation (Martínez-Rosell, Giorgino, & De Fabritiis, 2017), neural network-based virtual screening (Skalic, Martínez-Rosell, Jiménez, & De Fabritiis, 2019), binding pocket prediction (Jiménez, Doerr, Martínez-Rosell, Rose, & De Fabritiis, 2017), and ligand properties (Skalic, Varela-Rial, Jiménez, Martínez-Rosell, & De Fabritiis, 2019). Koes et al. developed ligand-based virtual screening web services, Pharmit (Sunseri & Koes, 2016), and ZINCPharmer (Koes & Camacho, 2012). Although there seems to be some overlap between these web services and ezCADD, the underlying computational approaches and implementation and are completely different. For

example, we plan to implement GPU/CPU hybrid methods for ligand-based virtual screening. Unlike PlayMolecule.org which utilizes a stripped-down version of NGL Viewer (A. S. Rose & Hildebrand, 2015) for 3D molecular visualization, we took the opposite approach by building new CADD applications and functions on top of the original NGL. This design allows users to take full advantages of all NGL's molecular visualization features and high-quality image rendering (Tao et al., 2019).

To date, we have developed eight fundamental CADD applications for ezCADD platform: 1) small-molecule docking (ezSMDock); 2) binding site detection (ezPocket); 3) protein-protein docking (ezPPDock); 4) 2D/3D protein-ligand interactions visualizer (ezLigPlot); 5) high throughput virtual screening (ezHTVS); 6) *de novo* lead optimization (ezGrow); 7) drug target and poly-pharmacology identification (ezPocketSearch); 8) cross-database molecule search (ezTargetSearch).

We will also show the impact of ezCADD on students who mostly had no prior computational background through an in-class modeling exercise followed by a survey of user experience (Tao et al., 2019).

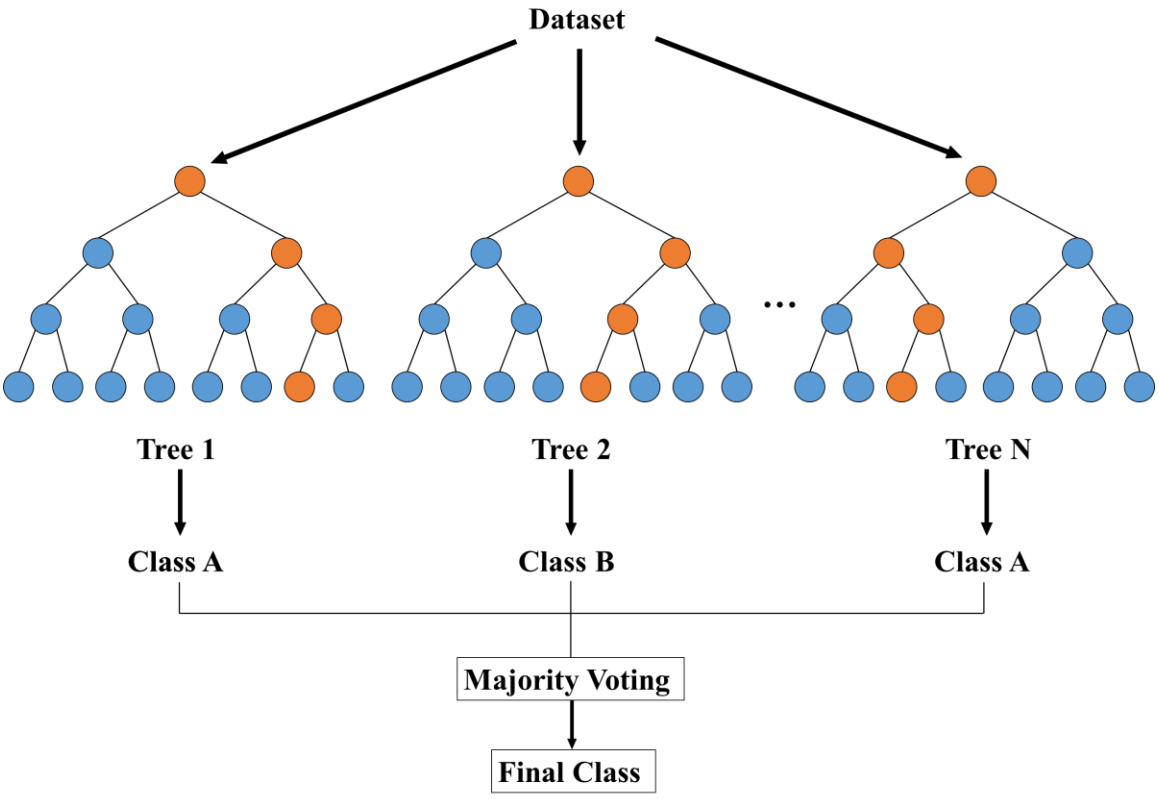


Figure 1 Decision process for random forest workflow. Yellow circles represent decision path of the tree.

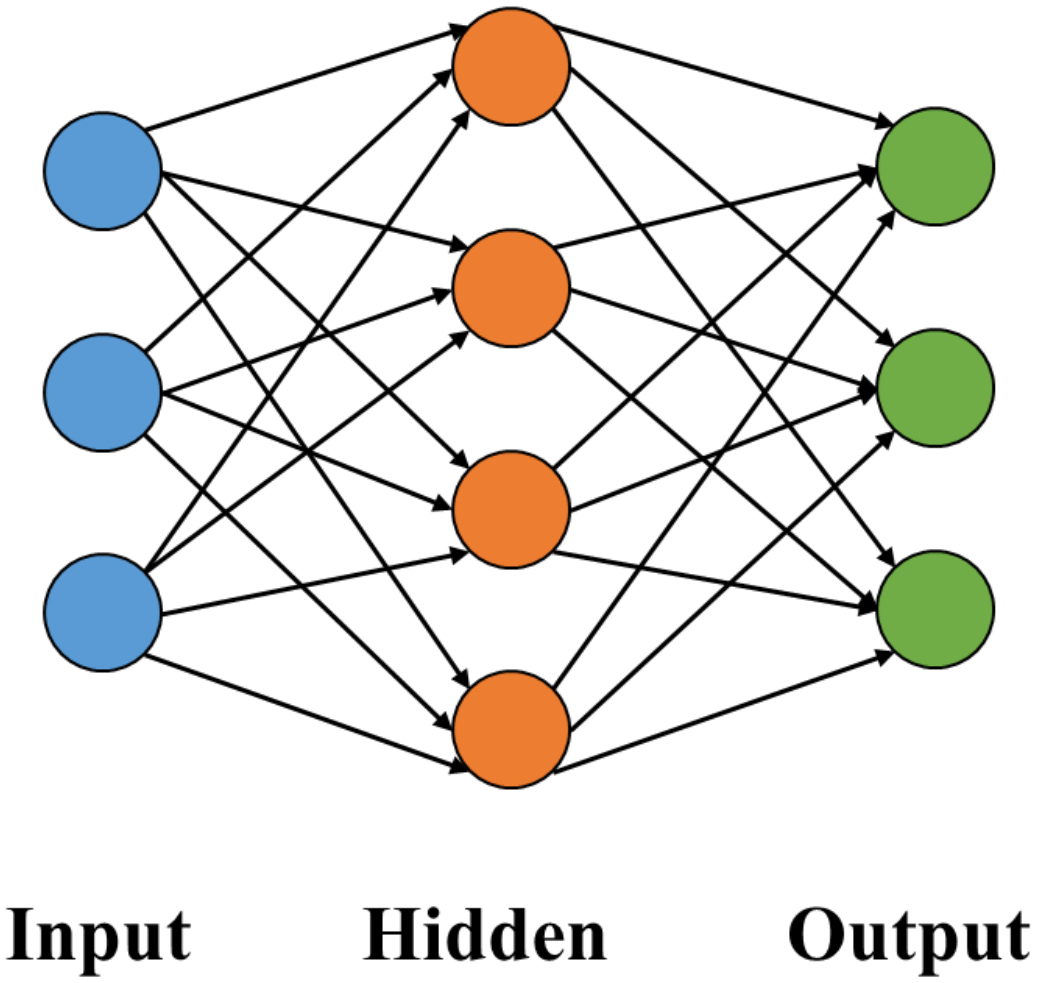


Figure 2 Schematic of artificial neural network. The blue nodes represent input layer. The orange nodes represent hidden layer. The green nodes represent output layer. Arrows between two nodes represent connections between different layers.

I.2 ezCADD Application Implementation and Graphic User Interface (GUI)

Four simple design concepts, easy, quick, user-friendly, and 2D/3D visualization-enabled, were maintained throughout ezCADD development. To realize these design goals, we embraced the NGL Viewer, a versatile WebGL-based (Marrin, 2011) application, as our 3D molecular visualizer. We added a JavaScript-based 2D molecular visualizer and integrated a set of high-performance free/open-source software with the 2D/3D molecular visualizers through a combination of multiple client-server orientated programming languages (shown in Figure 3). ezCADD is designed to be system agnostic and accessible from anywhere by any computers or tablets with an up-to-date web browser that has the latest JavaScript support. ezCADD has been successfully tested on the latest version of Firefox, Chrome, and Safari web browsers.

Frontend Implantation

ezCADD adopts the original GUI layout of NGL Viewer, with the center WebGL-based 3D window dedicated to molecular visualization and the right navigation panel dedicated to visualization controls. CADD applications are added to the top menu bar. A left navigation panel is added to display computational results. It is noteworthy that both left and right navigation panels can be flexibly minimized or resized by double-clicking or dragging the panel edge to make more space available for the 3D visualization window. ezCADD has a floating window GUI design, i.e., invoked windows and menus can be overlaid and freely moved around; closed windows and menus can be re-invoked without interrupting the current task. Imported molecular files or representations are accessible until they are updated during the same session.

Backend Implantation

ezCADD has implemented a job tracking program, which is capable of multi-tasking job recording for millions of times in parallel without any errors. The job tracker is written in C++ programming language using the C++11 standard (sample C++ source code is shown in Figure 4). Line 1 to 9 of Figure 4 is the description of the usage and purpose of this program. All the required libraries will be included from line 12 to 23. This program has defined 3 different modes for job status, which are SUBMIT, START, and END. In lines 29 to 33 of Figure 4, the count of arguments will be compared to the number 7 to further proceed. The current system time will be recorded and held by the time_t variable "now_Time" in the line 34. The user IP, name of the application, and job ID will be combined and left-aligned to an output stream class variable "str_uni" in lines 42 to 44. The log file under the user-specified directory will be checked in the function "log_FILE" (line 45 of Figure 4, and Figure 7). Then the job status will be recorded in the function "job_STATUS" (line 46 of Figure 4, and Figure 5 and Figure 6).

In the function of log file checking "log_FILE" (source code is shown in Figure 7), a pointer variable of constant character "log_name" is defined to receive the name of the log file in line 1. An input file stream class object variable "non_log" is defined and initialized with the name of "log_name" in line 3, and an output file stream class object variable "log" is defined in line 4. If the file "non_log" does not exist (line 6), then the output file "log" will be open (line 7) and initialized with the file information and title line (lines 8 to 17). Then the output file will be closed properly in line 18. This operation will not be executed if the log file already exists.

In the function of job status recording “job_STATUS” (source code is shown in Figure 5 and Figure 6), the references of constant string “folder”, “job”, “id”, “uni”, and “de_note”, the pointer of constant string “log_name”, and the time class “t” will be received (line 1). If the string variable “job” is “SUBMIT”, then the information of the current job will be written to the log file (lines 9 to 19). If the string variable “job” is “START”, then the lock for the file will be checked, and the job information will be written to the line with the unique job ID (line 19 to 45). If the string variable “job” is “END”, then the execution time of the job will be calculated and written to the line with the unique job ID in the log file (lines 45 to 72).

This job tracking system has provided an essential function for the ezCADD job submission and control. With the integration of the job tracking program, all the backend jobs will be recorded in a well-formatted log file. The log file is currently stored and maintained in the directory “/host/daemon/phptemp/log”.

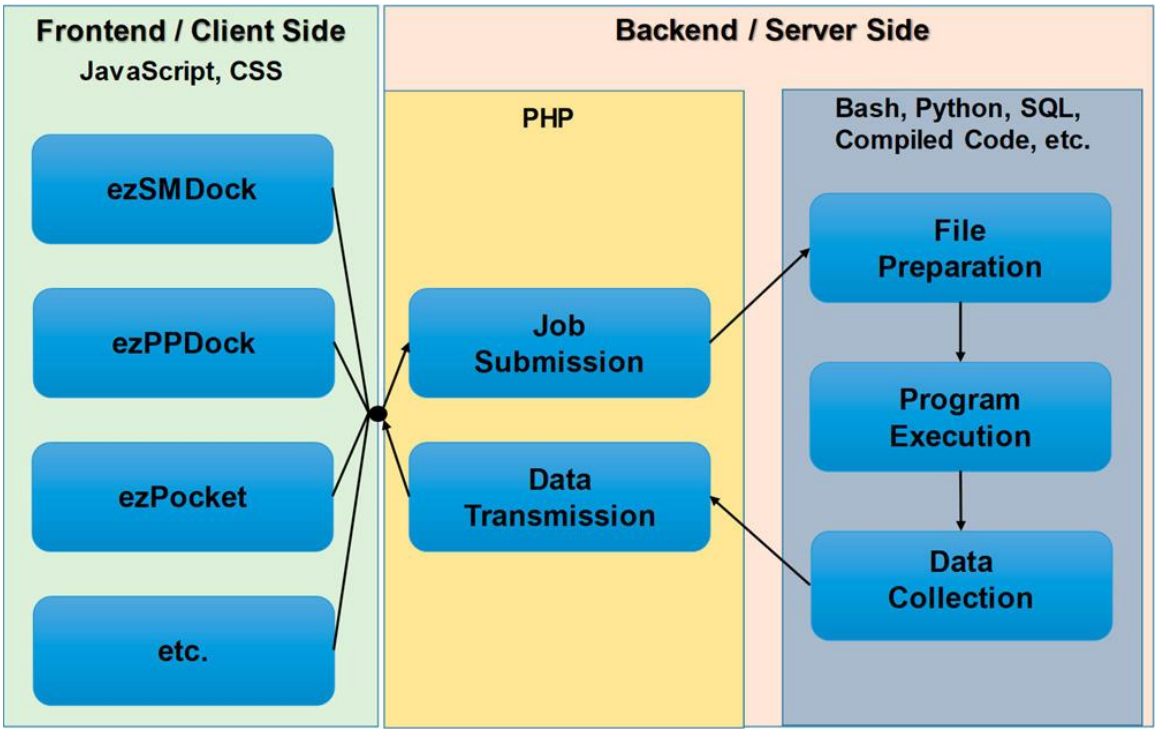


Figure 3 ezCADD web service implementation.

```cpp
/*
 * Rule:
 *     ./jobTracker STATUS IP App JobID Time Duration Note LogPath
 * Example:
 *     ./jobTracker SUBMIT 192.168.1.1 ezSMDock 10000000000000000000000000000001 DEBUG .
 *     ./jobTracker START 192.168.1.1 ezSMDock 10000000000000000000000000000001 DEBUG .
 *     ./jobTracker END 192.168.1.1 ezSMDock 10000000000000000000000000000001 DEBUG .
 * File Format:
 * IP(17) App(16) JobID(35) TimeSubmit(13) TimeStart(13) TimeEnd(13) Duration(13) Note(8)
 */

#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <string>
#include <ctime>
#include <algorithm>
#include <vector>
#include <cstdio>
#include <chrono>
#include <thread>
#include <cstring>

using namespace std;

int main(int argc, char *argv[])
{
    if (argc < 7){
        cout << "Example: "<< argv[0]
             << " [(Status) SUBMIT/START/END] [(IP) 192.168.1.1] [(App) ezSMDock] [(ID)
             10000000000000000000000000000001] [DEBUG/Anynotes] [logPATH]\n";
        return 1;
    }
    time_t now_Time = time(0);
    string str_out, job_status, job_id;
    ostringstream str_uni;
    string de_note = argv[5];
    job_status = argv[1];
    job_id = argv[4];
    string folder_name = argv[6];
    const char *log_name = str2arr(folder_name, "/jobTracker.log");
    str_uni << left << setw(17) << argv[2]
            << left << setw(16) << argv[3]
            << left << setw(35) << argv[4];
    log_FILE(log_name);
    job_STATUS(folder_name, log_name, job_status, job_id, str_uni.str(), now_Time,
    de_note);

    return 0;
}
```

Figure 4 Sample CPP source code of job tracker.

File location of C++ source code for job tracker:

~/Desktop/test_folder/C_test/IPTracker/iptracker.cc

Compilation command (using ISO C++ 2011 standard):

/usr/bin/g++-4.8 -std=c++11 -o jobTracker.out iptracker.cc

```cpp
void job_STATUS(const string &folder, const char *log_name, const string &job,
                const string &id, const string &uni, time_t t, const string &de_note)
{
    ifstream fi;
    const char *tmp_log = str2arr(folder, "/tmp_iptracker.log");
    const char *tmp_lock = str2arr(folder, "/tmp_iptracker.lock");
    ofstream tmp_fo, log;
    string tmp_str, r_out;
    if (job == "SUBMIT"){
        log.open(log_name, ios::app);
        log << uni
            << left << setw(13) << t
            << left << setw(13) << "0000000000"
            << left << setw(13) << "0000000000"
            << left << setw(13) << "00000"
            << left << setw(8)  << de_note
            << endl;
        log.close();
    } else if (job == "START"){
        lock_STATUS(tmp_lock, 1);
        fi.open(log_name);
        tmp_fo.open(tmp_log);
        while (getline(fi, r_out)){
            size_t pos = r_out.find(id);
            if (pos != string::npos) {
                vector<string> vs = split(r_out);
                vs[4] = to_string(t);
                tmp_fo << left << setw(17) << vs[0]
                       << left << setw(16) << vs[1]
                       << left << setw(35) << vs[2]
                       << left << setw(13) << vs[3]
                       << left << setw(13) << vs[4]
                       << left << setw(13) << "0000000000"
                       << left << setw(13) << "00000"
                       << left << setw(8)  << de_note
                       << endl;
            } else {
                tmp_fo << r_out << '\n';
            }
        }
        fi.close();
        tmp_fo.close();
        rename(tmp_log, log_name);
        remove(tmp_lock);
    } else if (job == "END"){
        lock_STATUS(tmp_lock, 1);
        fi.open(log_name);
        tmp_fo.open(tmp_log);
        while (getline(fi, r_out)){
            size_t pos = r_out.find(id);
            if (pos != string::npos) {
                vector<string> vs = split(r_out);
                vs[5] = to_string(t);
                long v6 = difftime(stol(vs[5]), stol(vs[4]));
```

Figure 5 Sample CPP source code of job status fuction (part 1 of 2) in job tracker program.

```
55                  tmp_fo << left << setw(17) << vs[0]
56                         << left << setw(16) << vs[1]
57                         << left << setw(35) << vs[2]
58                         << left << setw(13) << vs[3]
59                         << left << setw(13) << vs[4]
60                         << left << setw(13) << vs[5]
61                         << left << setw(13) << v6
62                         << left << setw(8)  << de_note
63                         << endl;
64              } else {
65                  tmp_fo << r_out << '\n';
66              }
67          }
68          fi.close();
69          tmp_fo.close();
70          rename(tmp_log, log_name);
71          remove(tmp_lock);
72      } else {
73          cout << "ERROR: argv[1] must be SUBMIT, START, or END." << endl;
74      }
75  }
```

Figure 6 Sample CPP source code of job status fuction (part 2 of 2) in job tracker program.

The file location of C++ source code for job tracker:

~/Desktop/test_folder/C_test/IPTracker/iptracker.cc

```
1   void log_FILE(const char *log_name)
2   {
3       ifstream non_log(log_name);
4       ofstream log;
5       // If log file doesn't exist, create one.
6       if (!non_log){
7           log.open(log_name);
8           log << "#This file logs applications usage of ezCADD website, generated by
            iptracker.\n"
9               << left << setw(17) << "IP_Address"
10              << left << setw(16) << "AppName"
11              << left << setw(35) << "JobID (DirectoryName)"
12              << left << setw(13) << "TimeSubmit"
13              << left << setw(13) << "TimeStart"
14              << left << setw(13) << "TimeEnd"
15              << left << setw(13) << "Duration(s)"
16              << left << setw(8)  << "Note"
17              << endl;
18          log.close();
19      }
20  }
```

Figure 7 CPP source code of log file check function.

The file location of C++ source code for job tracker:

~/Desktop/test_folder/C_test/IPTracker/iptracker.cc

I.3 ezSMDock: Small Molecule Docking

Frontend Implementation

In this web application, the receptor structure can be imported by either uploading a Protein Data Bank (PDB) (P. W. Rose et al., 2010) file or entering a PDB ID. ezSMDock provides a variety of options for users to import the ligand structure: (1) a 3D ligand file in PDB, MOL2, or SDF/MOL format; (2) a 2D ligand file in SDF/MOL format; (3) SMILES string; and (4) InChI string, all of which can be easily obtained and downloaded from public chemical/drug databases such as PubChem (Bolton, Wang, Thiessen, & Bryant, 2008), ChEMBL (Gaulton et al., 2012), RSCB/PDB (P. W. Rose et al., 2010), DrugBank (Wishart et al., 2006), and Wikipedia (https://www.wikipedia.org/) or Google (https://www.google.com/) searches. The combination of PDB ID, SMILES, and InChI allows users to easily perform a docking experiment without the need for downloading and uploading files. For medicinal chemists who work on de novo drug design, ezSMDock enables them to draw and modify molecular structures of interest using PubChem Sketcher (Ihlenfeldt, Bolton, & Bryant, 2009), automatically generates SMILES/InChI strings of the new structures as ligand input, and quickly docks them into receptor binding site. This makes ezSMDock an efficient tool for medicinal chemists to rapidly enumerate diverse structural possibilities and predict associated binding affinity changes for exploring structure activity relationships (SARs) and formulating hypotheses.

The input ligand file or representation is converted to a MOL2 file using Unicon (Sommer et al., 2016), which calculates the best protonation and tautomer state at the physiological pH. An InChI string is converted to SMILES string using OpenBabel (O'Boyle et al., 2011) and fed into Unicon for MOL2 file generation. The MOL2 file is

then converted to a PDBQT file using MGLTools (Forli et al., 2016). The receptor PDB file is cleaned up using BASH and regular expression before it is converted to a PDBQT file using MGLTools. Both the receptor and ligand PDBQT files are used as the input files for AutoDock Vina (Trott & Olson, 2010) or Smina (Koes, Baumgartner, & Camacho, 2013) docking on the server-side. Both programs are highly efficient and considered one of the best open-source small-molecule docking code. The XYZ coordinates of the docking box center can be entered manually or automatically populated by clicking the "User Ligand Center" button when a 3D ligand file is provided. If users choose to manually enter the center coordinates, the data can be obtained from ezPocket (see section I.4), which provides the center coordinates of all predicted binding pockets. In addition, users can set up the appropriate box size. The docking box is automatically updated with user input. Upon setting up the parameters, a docking job is submitted for execution on the server side by clicking the "Start" button.

An ezSMDock job typically takes a few seconds. Upon completion, the docked poses and predicted scores are displayed in the left navigation panel. Users can click on each pose to inspect their binding modes in the 3D visualization window. A pop-up window is also launched to display the 2D structure of the input ligand. The 2D molecular visualization is implemented using Kekule.js open-source JavaScript library (Jiang, Jin, Dong, & Chen, 2016). The rapid 2D and 3D molecular visualization in combination provide users with ease and flexibility to compare multiple ligands after docking. A zip file containing (1) user receptor and ligand inputs, (2) all docked poses in an SDF file, and (3) all docking scores in a tab-delimited TSV file, is also available for download and further analysis. It is recommended that users perform re-docking of

ligands with known experimentally determined bound coordinates to validate the docking parameters before docking other molecules.

ezSMDock currently allows docking one ligand at a time for the purpose of resource sharing and abuse prevention. A new high throughput virtual screening web application (ezHTVS) is under development and the beta version is added to ezCADD for users to screen millions of compounds within minutes.

Backend Implementation

In the backend of ezSMDock, a PHP script is in charge of receiving parameters from front-end JavaScript, processing parameters and passing them to Bash scripts, and return the results to frontend JavaScript.

As shown in Figure 8, all the parameters will be passed from JavaScript to PHP script by the variable "$_POST". In line 3, a string variable "$version_no" controls the current version of the backend script. In line 5, a string variable "$receptor_format" will hold the format of the receptor, as PDB file or PDB ID. In line 19, the variable "$ligand_format" will hold the ligand format, as 3D structure file, 2D structure file, SMILES string, or InChI String. The center of the docking box will be received by the variables "$co_x", "$co_y", and "$co_z" in lines 21 to 23, and the size of each side of the docking box will be received by the variables "$s_x", "$s_y", and "$s_z" in lines 25 to 27. The docking program will be received and defined in lines 8 to 15. All of those parameters will be processed and passed to Bash scripts to perform small molecule docking.

The sample Bash script for preparation of receptor and ligand and execution of molecular docking is shown in Figure 9. All the parameters from the PHP script will be received and defined in lines 3 to 15. Multiple processors will be used to accelerate the execution of molecular docking in line 16. In lines 18 to 27, the ligand will be processed to generate a 3D structure in the mol2 file. Then the 3D ligand structure will be prepared in pdbqt format for molecular docking in line 30. The 3D receptor structure will also be prepared in pdbqt format for molecular docking, in lines 33 and 34. All the docking parameters will be written to a configure file in lines 37 and 38. The final molecular docking will be performed in lines 40 to 48, depending on the docking program user selected in the frontend.

The entire backend process of ezSMDock usually will be finished within one minute. The docking results will be collected and returned to the JavaScript, and the docking poses will be shown in the ezCADD main window.

```php
<?php

$version_no = "four";

$receptor_format = $_POST["SM_RF"];
$receptor = "Receptor.pdb";

$method = $_POST["SM_Method"];
if ($method == "vina") {
  $ligand_out = "ezSMDock_Vina.sdf";
  $score_out = "ezSMDock_Vina.tsv";
} elseif ($method == "smina"){
  $ligand_out = "ezSMDock_Smina.sdf";
  $score_out = "ezSMDock_Smina.tsv";
}

$zip_out = 'ezSMDock_result.zip';
//$ligand_format has "2D", "3D", "SMILES", and "InChI" four types.
$ligand_format = $_POST["SM_Format"];
$SM_string = $_POST["SM_String"];
$co_x = $_POST["CoordinateX"];
$co_y = $_POST["CoordinateY"];
$co_z = $_POST["CoordinateZ"];

$s_x = $_POST["SizeX"];
$s_y = $_POST["SizeY"];
$s_z = $_POST["SizeZ"];

?>
```

Figure 8 Sample PHP source code for ezSMDock parameter receiving.

The file location of PHP source code for ezSMDock parameter receiving:

/host/apps/ezCADD/php2/smdock.php

```
#!/bin/bash

user_receptor=$1
user_ligand=$2
user_format=$3
user_method=$4
c_x=$5
c_y=$6
c_z=$7

s_x=$8
s_y=$9
s_z=${10}
out_ligand=${11}
out_score=${12}
no_cpu=8

if [ $user_format == "3D" ]
then
    # Remove hydrogen atoms from original file and re-add hydrogen atoms to mol2 file
    (Because unicon does not support pdb as output type).
    /host/apps/anaconda3_smol/bin/babel -f1 -l1 -d $user_ligand -opdb user_query_noH.pdb
     2>/dev/null
    /host/daemon/bin/Unicon -i user_query_noH.pdb -o user_query_addH.mol2 --generate=1
    --verbosity=0
elif [ $user_format == "SMILES" ] || [ $user_format == "InChI" ] || [ $user_format ==
"2D" ]
then
    /host/apps/anaconda3_smol/bin/babel -f1 -l1 -d $user_ligand -osmi user_query.smi 2>/
    dev/null
    /host/apps/anaconda3_smol/bin/babel --gen3D -h -ismi user_query.smi -omol2
    user_query_addH.mol2 2>/dev/null
fi

# Generate pdbqt file for ligand.
/host/apps/mgltools_x86_64Linux2_1.5.6/MGLToolsPckgs/AutoDockTools/Utilities24/
prepare_ligand4.py -l user_query_addH.mol2 -o user_query_addH.pdbqt &>/dev/null

# Prepare receptor pdbqt file.
grep -E "^ATOM" $user_receptor > user_receptor_processed.pdb
/host/apps/mgltools_x86_64Linux2_1.5.6/MGLToolsPckgs/AutoDockTools/Utilities24/
prepare_receptor4.py -A hydrogens -U nphs_lps_waters_nonstdres_deleteAltB -r
user_receptor_processed.pdb -o user_receptor.pdbqt &>/dev/null

# Put all the parameters into conf.txt file.
printf "receptor = user_receptor.pdbqt\nligand = user_query_addH.pdbqt\ncpu =
$no_cpu\n\n" > conf.txt
printf "center_x = $c_x\ncenter_y = $c_y\ncenter_z = $c_z\n\nsize_x = $s_x\nsize_y =
$s_y\nsize_z = $s_z\n" >> conf.txt

# Docking.
if [ $user_method == "vina" ]
then
    /host/apps/autodock_vina_1_1_2_linux_x86/bin/vina --config conf.txt --out
    user_query_docked.pdbqt --log user_query_docked.log &>/dev/null
    /host/apps/anaconda3_smol/bin/babel -ipdbqt user_query_docked.pdbqt -osdf
    $out_ligand 2>/dev/null
elif [ $user_method == "smina" ]
then
    /host/apps/smina/smina.static2017-11-09 --config conf.txt --out $out_ligand --log
    user_query_docked.log &>/dev/null
fi

```

Figure 9 Sample Bash script for ezSMDock.

File location of Bash script for ezSMDock: /host/apps/ezCADD/SMDock/dockone.sh

I.4 ezPocket: Binding Site Prediction

Frontend Implementation

In this web application, the receptor structure can be imported by either uploading a PDB file or entering a PDB ID. ezPocket provides users with a consensus approach to detect plausible receptor binding pockets. Three popular cavity detection programs are currently offered: (1) fconv (Neudert & Klebe, 2011), (2) fpocket2 (Le Guilloux, Schmidtke, & Tuffery, 2009), and (3) fpocket3 (Le Guilloux et al., 2009; Schmidtke, Le Guilloux, Maupetit, & Tuffery, 2010). fconv uses Delaunay triangulation with weighted points to detect cavities whereas fpocket2 and fpocket3 rely on fast Voronoi tessellation. Upon setting up the parameters, a job is submitted for execution on the server-side by clicking the "Start" button.

An ezPocket job typically takes a few seconds. Upon completion, the detected binding cavities, their associated cavity center XYZ coordinates, and cavity volumes are displayed in the left navigation panel. All detected pockets are shown in a gray wireframe representation in the 3D visualization window. When users select a pocket in the left navigation panel, its corresponding wireframe is highlighted in cyan in the 3D visualization window. Users can use the consensus results predicted by multiple programs in conjunction with their biochemical intuition and knowledge of the receptor to determine the most plausible cavity for small-molecule binding. As described earlier, once users determine a binding cavity, the center XYZ coordinates of the cavity can be entered into ezSMDock to set up a molecular docking job. A zip file containing (1) user receptor input, (2) all detected pockets in a 3D multi-model MOL2 file, and (3) the XYZ

center coordinates and volumes of all detected pockets in a tab-delimited TSV file, is also available for download and further analysis.

Backend Implementation

The backend of the ezPocket application consists of a PHP script (sample code shown in Figure 10) embedded with a Bash script (sample code shown in Figure 11).

In the PHP script of ezPocket as shown in Figure 10, the parameters will be received from JavaScript in lines 7 to 10. In lines 11 to 20, the output files will be defined based on the string variable "$method" from user selection. The receptor file will be moved or downloaded from the RCSB PDB website to the temporary working directory, in lines 24 to 28. Then the parameters will be passed to the Bash script to be executed, in line 29. The final results will be zipped in line 30 and returned to the front-end JavaScript.

As shown in Figure 11, the Bash script of ezPocket receives the parameters from the PHP script in lines 4 to 7. The parameters for pocket2 and pocket3 are defined in lines 12 to 19. If the method fconv is selected (from lines 21 to 34), then the fconv program will be executed to detect all the possible pockets in the receptor. All the pocket information will be collected and written to a TSV file. If the method pocket2 or pocket3 is selected (from line 35 to 50), then the corresponding program will be executed on the receptor to search for all the possible cavities. All the center coordinate of pockets will be calculated, collected, and written to the TSV file.

After the backend execution of ezPocket, the coordinates will be returned to the JavaScript. And the pockets will be shown in the main window of ezCADD with a meshed surface representation.

```php
<?php

$version_no = "three";
$userIP = getRealIpAddr();

// The format of receptor could be "file" or "pdb"
$receptor_format = $_POST["P_RF"];
$receptor = "Receptor.pdb";

$method = $_POST["Pocket_Method"];
if ($method == "fconv"){
  $pocket_out = "ezPocket_fconv.mol2";
  $coordinate_out = "ezPocket_fconv.tsv";
} elseif ($method == "fpocket2"){
  $pocket_out = "ezPocket_fpocket2.mol2";
  $coordinate_out = "ezPocket_fpocket2.tsv";
} elseif ($method == "fpocket3"){
  $pocket_out = "ezPocket_fpocket3.mol2";
  $coordinate_out = "ezPocket_fpocket3.tsv";
}

$zip_out = "ezPocket_result.zip";

if ($receptor_format == "file"){
  move_uploaded_file($_FILES["Receptor"]["tmp_name"],$receptor);
} elseif ($receptor_format == "pdbid"){
  shell_exec('wget -O '.$receptor.' https://files.rcsb.org/view/'.$_POST["Receptor"].
  '.pdb 2>/dev/null');
}
shell_exec("/host/apps/ezCADD/Pocket/poktek.sh ".$receptor." ".$method." ".$pocket_out.
" ".$coordinate_out);
shell_exec('/usr/bin/zip -rq '.$zip_out.' '.$receptor.' '.$pocket_out.' '.
$coordinate_out);

?>
```

Figure 10 Sample PHP source code of ezPocket.

The file location of PHP source code of ezPocket:

/host/apps/ezCADD/php2/poktek.php

```
#!/bin/bash

# Parameters for all methods
user_receptor="target.pdb"
user_method=$2
user_mol2=$3
user_tsv=$4
po_ctrl=0
printf "Name\tX\tY\tZ\tVol(A^3)\n" > $user_tsv

# Parameters for fpocket(2&3)
LD_LIBRARY_PATH=/host/apps/amber16/lib
if [ $user_method == "fpocket2" ]
then
    vol_word="Real volume"
elif [ $user_method == "fpocket3" ]
then
    vol_word="Pocket volume (Monte Carlo)"
fi

if [ $user_method == "fconv" ]
then
    /host/apps/fpocket/fconv -pa2 $user_receptor &>tmp.log
    for i in $(ls -v *.mol2)
    do
        po_name="Pocket"$po_ctrl
        c_x=$(grep -A10000 "@<TRIPOS>ATOM" $i | grep -v "@<TRIPOS>ATOM" | awk '{print $3}' | awk '{ci+=$1}END{if(NR>0) print ci/NR}')
        c_y=$(grep -A10000 "@<TRIPOS>ATOM" $i | grep -v "@<TRIPOS>ATOM" | awk '{print $4}' | awk '{ci+=$1}END{if(NR>0) print ci/NR}')
        c_z=$(grep -A10000 "@<TRIPOS>ATOM" $i | grep -v "@<TRIPOS>ATOM" | awk '{print $5}' | awk '{ci+=$1}END{if(NR>0) print ci/NR}')
        c_vol=$(grep -B1 $i tmp.log | grep -v $i | cut -f2 -d= | awk '{print $1}')
        cat $i >> $user_mol2
        printf "$po_name\t$c_x\t$c_y\t$c_z\t$c_vol\n" >> $user_tsv
        po_ctrl=$(expr $po_ctrl + 1)
    done
elif [ $user_method == "fpocket2" ] || [ $user_method == "fpocket3" ]
then
    /host/apps/fpocket/$user_method -f $user_receptor &>tmp.log
    for i in $(ls -v $fpo_folder"/"*_atm.pdb)
    do
        po_name="Pocket"$po_ctrl
        tmp_mol2=$po_name".mol2"
        c_x=$(egrep '^HETATM|^ATOM' $i | cut -c 31-38 | awk '{ci+=$1}END{if(NR>0) print ci/NR}')
        c_y=$(egrep '^HETATM|^ATOM' $i | cut -c 39-46 | awk '{ci+=$1}END{if(NR>0) print ci/NR}')
        c_z=$(egrep '^HETATM|^ATOM' $i | cut -c 47-54 | awk '{ci+=$1}END{if(NR>0) print ci/NR}')
        c_vol=$(grep "$vol_word" $i | cut -f2 -d: | awk '{print $1}')
        /host/apps/anaconda3_smol/bin/babel --title $po_name -ipdb $i -omol2 $tmp_mol2 &>/dev/null
        cat $tmp_mol2 >> $user_mol2
        printf "$po_name\t$c_x\t$c_y\t$c_z\t$c_vol\n" >> $user_tsv
        po_ctrl=$(expr $po_ctrl + 1)
    done
fi

```

Figure 11 Sample Bash script for ezPocket

File location of Bash script for ezPocket:

/host/apps/ezCADD/Pocket/poktek.sh

I.5 ezPPDock: Protein-Protein Docking

Frontend Implementation

In this web application, both receptor and ligand structures can be imported by either uploading a PDB file or entering a PDB ID. If users feel that the HETATM entries (water, metal ions, glycosylated sugars, etc.) surrounding the receptor should be taken into account during the protein−protein docking, they are provided an option to keep them. Users need to specify the chain IDs of both receptor and ligand that are involved in the protein-protein interactions. If the users have biochemical information indicating some residues do not belong to the actual binding interface, they can specify any receptor and/or ligand residues to be blocked from the docking experiment. All blocked residues are highlighted in solid green surface representation in the 3D visualization window. Upon setting up the parameters, a job is submitted for execution on the server-side by clicking the "Start" button. MegaDock 4.0 (Ohue et al., 2014) is used as the docking engine because of its ultra-high-performance and the support of GPU/CPU hybrid computing.

An ezPPDock job typically takes a few seconds. Upon completion, the top 2000 docked poses and their associated scores are displayed in the left navigation panel. Users can inspect the docked poses in the 3D visualization window by selecting them in the left navigation panel. ezPPDock also takes advantage of NGL's molecular dynamics trajectory support by loading all 2000 docked poses in a compressed DCD trajectory file to the right control panel. All docked poses can be animated and viewed in the form of automatic movie playback. A zip file containing (1) user receptor and ligand inputs, (2) 2000 docked poses in a DCD trajectory file, and (3) docking scores associated with all

docked poses in a tab-delimited text output file is also available for download and further analysis.

Backend Implementation

The backend operation of ezPPDock is mainly based on the PHP script (sample code shown in Figure 12 and Figure 13). In lines 6 to 10, the essential input and output files are defined. The JSON strings of receptor and ligand received from frontend JavaScript are decoded to PHP variables in lines 11 and 12, respectively. The format of the trajectory file is received in line 14 and the name of the output trajectory file is defined in line 15.

If the format of the receptor is “file”, then the receptor file will be uploaded to the current working directory (as shown in lines 18 to 20). If the format of the receptor is “pdbid”, then the receptor file will be downloaded from the RCSB PDB website (as shown in lines 20 to 22). If the format of the ligand is “file”, then the ligand file will be uploaded to the current working directory (as shown in lines 23 to 25). If the format of the ligand is “pdbid”, then the ligand PDB file will be downloaded from the RCSB PDB website (as shown in lines 25 to 27).

For the receptor structure, if the user unselected “KeepHETATM” as default, only protein atoms will be retained without heteroatoms of amino acids, in line 30 to 34. If the “KeepHETATM” option is selected by the user, then both protein atoms and heteroatoms will be kept to the receptor PDB file in lines 34 to 38. For the ligand structure, only atoms of protein will be retained in line 39.

When the receptor or ligand contains more than one chain, the chain(s) of the protein will be selected and written to a new file (as shown in lines 43 to 46). When the user chose to exclude some residues for the protein-protein docking, the excluded amino acid residues will be blocked as “BLK” in the structure file for both receptor and ligand proteins (as shown in lines 49 to 78).

The protein-protein docking will be performed with the final prepared receptor protein file and ligand-protein file, by using multiple processors and GPUs enabled Megadock-GPU (as shown in line 80). All the docking results will be processed in line 81, zipped in line 82, and returned to the frontend JavaScript.

```php
<?php

$version_no = "three";
$userIP = getRealIpAddr();

$ori_receptor = "Receptor.pdb";
$ori_ligand = 'Ligand.pdb';
$out='ezPPDock_result.out';
$zip='ezPPDock_result.zip';

$info_rec = json_decode($_POST["Info_Rec"], true);
$info_lig = json_decode($_POST["Info_Lig"], true);

$traj_format = $_POST["Info_Output"];
$traj_out="$out.$traj_format";

// Upload receptor/ligand file or download receptor/ligand file from rcsb.
if ($info_rec["Format"] == "file"){
  move_uploaded_file($_FILES["Receptor"]["tmp_name"],$ori_receptor);
} elseif ($info_rec["Format"] == "pdbid"){
  shell_exec('wget -O '.$ori_receptor.' https://files.rcsb.org/view/'.$_POST["Receptor"].'.pdb 2>/dev/null');
}
if ($info_lig["Format"] == "file"){
  move_uploaded_file($_FILES["Ligand"]["tmp_name"],$ori_ligand);
} elseif ($info_lig["Format"] == "pdbid"){
  shell_exec('wget -O '.$ori_ligand.' https://files.rcsb.org/view/'.$_POST["Ligand"].'.pdb 2>/dev/null');
}

// Keep or discard HETATM for receptor .pdb file. Discard HETATM from ligand .pdb file.
if ($info_rec["KeepHETATM"] == "unselected"){
  $atm_pdb = 'Receptor_step1_no_HETATM.pdb';
  shell_exec('grep -E ^ATOM '.$ori_receptor.' > '.$atm_pdb);
  $receptor = $atm_pdb;
} elseif ($info_rec["KeepHETATM"] == "selected") {
  $atm_pdb = 'Receptor_step1_keep_HETATM.pdb';
  shell_exec('grep -E "^ATOM|^HETATM" '.$ori_receptor.' > '.$atm_pdb);
  $receptor = $atm_pdb;
}
shell_exec('grep -E ^ATOM '.$ori_ligand.' > Ligand_step1_no_HETATM.pdb');
$ligand = "Ligand_step1_no_HETATM.pdb";

// Keep the selected chain(s) for receptor and ligand .pdb files.
shell_exec('grep -E "^.{21}['.$info_rec["Sel_Chains"].']" '.$receptor.' > Receptor_step2_Sel_Chains.pdb');
$receptor = "Receptor_step2_Sel_Chains.pdb";
shell_exec('grep -E "^.{21}['.$info_lig["Sel_Chains"].']" '.$ligand.' > Ligand_step2_Sel_Chains.pdb');
$ligand = "Ligand_step2_Sel_Chains.pdb";

// Block the choose residues for receptor and ligand .pdb files.
$blk_rec = 1;
```

Figure 12 Sample PHP source code (part 1 of 2) of ezPPDock.

The file location of PHP source code of ezPPDock:

/host/apps/ezCADD/php2/ppd.php

```php
foreach ($info_rec["Residues"]["chains"] as $chain) {
  if ($chain["residues"] !="" and $chain["value"] !=" " ){
    $blk_curr_pdb = 'tmp_rec_BLK_'.$blk_rec.'.pdb';
    if ($blk_rec == 1){
      $blk_last_pdb = $receptor;
    } else {
      $blk_last_pdb = 'tmp_rec_BLK_'.$blk_last.'.pdb';
    }
    shell_exec('/host/apps/megadock-4.1.1/block '.$blk_last_pdb.' '.$chain["value"].' '.
    $chain["residues"].' > '.$blk_curr_pdb);
    $blk_last = $blk_rec;
    $blk_rec += 1;
    $receptor = $blk_curr_pdb;
  }
}
$blk_lig = 1;
foreach ($info_lig["Residues"]["chains"] as $chain) {
  if ($chain["residues"] !="" and $chain["value"] !=" " ){
    $blk_curr_pdb = 'tmp_lig_BLK_'.$blk_lig.'.pdb';
    if ($blk_lig == 1){
      $blk_last_pdb = $ligand;
    } else {
      $blk_last_pdb = 'tmp_lig_BLK_'.$blk_last.'.pdb';
    }
    shell_exec('/host/apps/megadock-4.1.1/block '.$blk_last_pdb.' '.$chain["value"].' '.
    $chain["residues"].' > '.$blk_curr_pdb);
    $blk_last = $blk_lig;
    $blk_lig += 1;
    $ligand = $blk_curr_pdb;
  }
}

shell_exec('export OMP_NUM_THREADS=12; env
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/host/apps/cuda8.0/lib64
/host/apps/megadock-4.1.1/megadock-gpu -R '.$receptor." -L ".$ligand." -o ".$out);
shell_exec('/host/apps/ezCADD/out2dcd.sh '.$receptor.' '.$ligand.' '.$out);
shell_exec('zip '.$zip.' '.$ori_receptor.' '.$ori_ligand.' '.$traj_out.' '.$out);

?>
```

Figure 13 Sample PHP source code (part 2 of 2) of ezPPDock.

The file location of PHP source code of ezPPDock:

/host/apps/ezCADD/php2/ppd.php

I.6 Case Study: Methylthioadenosine Nucleosidase

Methylthioadenosine nucleosidase (MTN) is an important enzyme in many pathogenic microbes, responsible for the catabolism of 5′-methylthioadenosine (MTA) and S-adenosylhomocysteine (SAH), two molecules involved in key microbial functions (Lee et al., 2005). Here we use the E. coli MTN as an example to demonstrate the applications of ezCADD in drug design.

ezPocket (Figure 14)

The E. coli MTN structure was imported from PDB ID 1Y6Q. Automatic cavity detection was performed using fconv, fpocket2, and fpocket3. There was a consensus result that matched the catalytic binding site of chain A, highlighted in cyan in the 3D visualization window (Figure 14). All three cavities predicted by the programs enclosed the bound inhibitor Methylthio-DADMe-Immucillin A (TDI). If we pretended that there was no known inhibitor inside the cavity, we could record one of three sets of reported center XYZ coordinates for the next molecule docking step.

ezSMDock (Figure 15)

The E. coli MTN structure was imported from PDB ID 1Y6Q. The SMILES or InChI string of inhibitor TDI were be obtained from PubChem (CID 656970). In this case, the InChI string was used as ligand input. The docking box center XYZ coordinates were taken from ezPocket, and the box size was set to 15 (Å). This re-docking experiment was performed using AutoDock Vina. Figure 15 shows that the highest scored pose was selected in the left navigation panel and displayed (ball and stick) on top

of the crystallographically bound TDI (licorice) in the 3D visualization window. AutoDock Vina correctly reproduced the experimental binding mode of TDI.

SAR and de novo *Drug Design (Figure 16)*

Here we used retrospective experimental data of two TDI derivatives to demonstrate how easily medicinal chemists can use ezCADD to study SAR and design novel compounds. First, the SMILES or InChI string of inhibitor TDI was copied and pasted to PubChem Sketcher for structural modification to create Benzylthio-DADMe-Immucillin A (DF9) and 4-Cl-phenylthio-DADMe-Immucillin A (4CT). Next, the automatically generated InChI strings of DF9 and 4CT were copied and pasted back to ezSMDock as ligand input for docking the congeneric ligands to MTN. Figure 16 shows that the docking results were displayed in the left navigation panel, the 2D structures of docked molecules were compared in the 2D visualization pop-up windows, and the top scored docked poses of TDI, DF9, and 4CT were overlaid in the 3D visualization window. Table 1 shows that the trend of the predicted docking scores was in good agreement with published experimental data (Gutierrez et al., 2007).

ezPPDock (Figure 17)

MTN is a homodimer. Modeling the protein-protein interactions at the dimer interface may offer new molecular insights into MTN enzymatic activity and potential inhibitor design that disrupts its dimerization and catalysis. Here we used ezPPDock to reconstruct MTN dimer from monomers. PDB ID 1Y6Q was imported, with chain B selected as the receptor. PDB ID 1NC1 was imported, with chain A selected as the ligand. Since we knew that residues 1-6 of the receptor and residues 10-16 of the ligand would not be involved in the protein-protein interactions, these residues were selected

(highlighted in the solid green surface) to be blocked during the docking experiment. This feature, analogous to the docking box used in small-molecule docking, may significantly improve protein-protein docking quality. Figure 17 shows that the top-scored pose was selected in the left navigation panel and displayed (yellow ribbon) with respect to the crystallographically bound monomer (red ribbon) in the 3D visualization window. MegaDock correctly reproduced the experimental assembly of MTN homodimer.

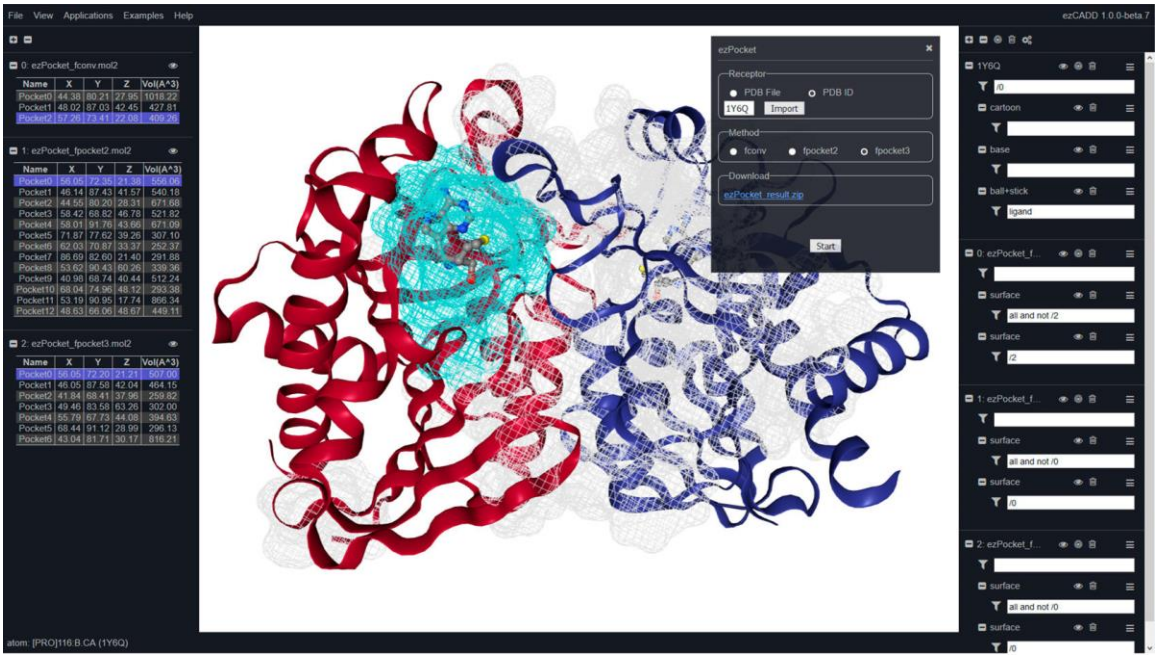


Figure 14 Consensus binding site detection using ezPocket (PDB ID 1Y6Q).

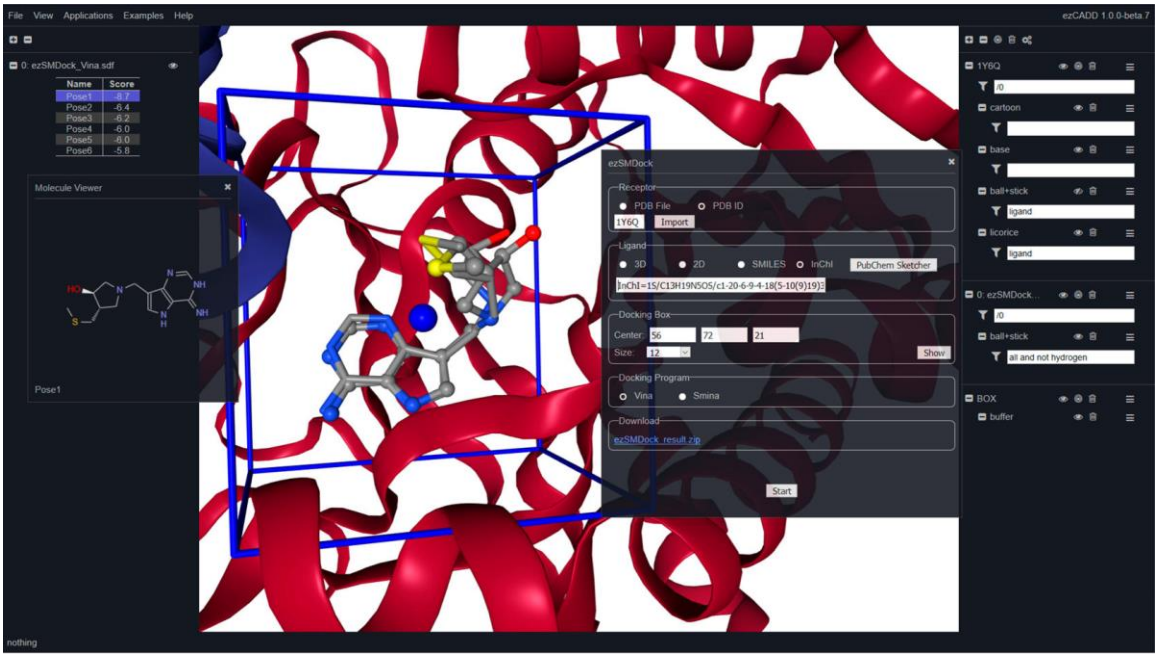


Figure 15 Redocking inhibitor TDI to E. coli MTN (PDB ID 1Y6Q) using ezSMDock.

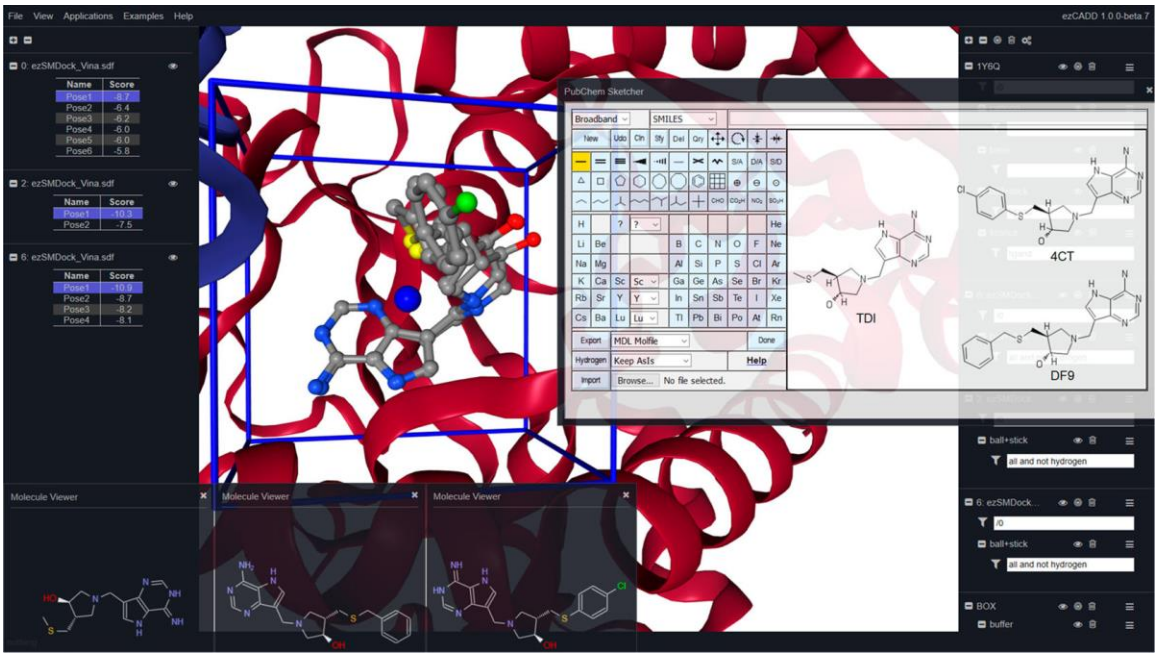


Figure 16 Computational SAR and de novo design using ezSMDock.

Table 1 Experimental inhibition constants and predicted docking scores of DADMe-Immucillin A derivatives on E.Coli MTN.

Inhibitor: DADMe-Immucillin A derivatives	**Experimental Kd (pM)**	**Docking Score**
methylthio-(TDI)	2	−8.7
benzylthio-(DF9)	0.46	−10.3
4-Cl-phenylthio-(4CT)	0.047	−10.9

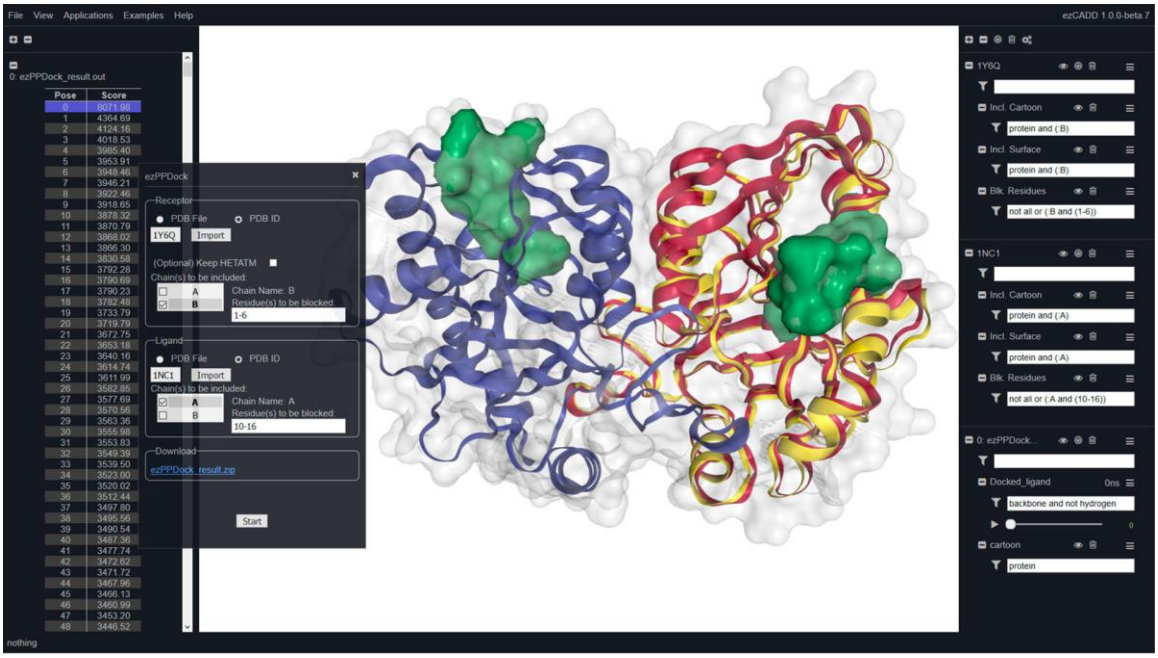


Figure 17 Re-constructing E. coli MTN homodimer (PDB ID 1Y6Q and 1NC1) using ezPPDock.

I.7 User Evaluation and Feedback

To assess user experience and the effectiveness of our implementation, we introduced ezCADD to 95 first-year pharmacy students as an active learning component in the Principles of Drug Action course. Before we started the ezCADD exercise, we collected the baseline data by asking the students about their level of experience with molecular modeling and visualization. We then loaded human β2 adrenergic receptor (PDB ID 3NYA) and bound drug alprenolol, a nonselective beta-blocker, into ezCADD. We used ezCADD's 2D and 3D molecular visualization to illustrate the structural and chemical features of the receptor and alprenolol. Next, we guided students to search PubChem for the SMILES or InChI representation of alprenolol and used them as the ligand input to re-dock the drug back to the receptor-binding site using ezSMDock. The web service handled 95 simultaneous docking jobs without an issue. All submitted jobs finished within minutes. We used ezCADD's 3D visualization again to explain molecular recognition and interactions between the receptor and alprenolol. Lastly, we followed up with five additional questions to gauge student experience. The results of the student survey are shown in Figure 18. This study received exempt status from Idaho State University Institutional Review Board (study number IRB-FY2019-38).

Our student survey data showed that, among the majority of students who participated, their experience level with molecular modeling and visualization was improved from zero or a little experience (64 % students) to some or good experience (79 % students). After the exercise, students with zero molecular modeling experience decreased to 0 % compared to a significant 32 % before the exercise (Figure 18A). At the end of the exercise, 99-100 % of students thought ezCADD enhanced their understanding

of drug and receptor structures as well as the concepts of drug-receptor binding and recognition (Figure 18B). 84 % of students fully completed the molecular docking experiment on their own whereas 14 % partially completed (Figure 18C). 88 % of students considered the tool to be easy and user-friendly (Figure 18D). The student feedback provided representative baseline data of ezCADD user experience for biomedical researchers with little or no computational background. It is clearly demonstrated that ezCADD is an easy, user-friendly, and powerful tool for both drug discovery research and STEM education.

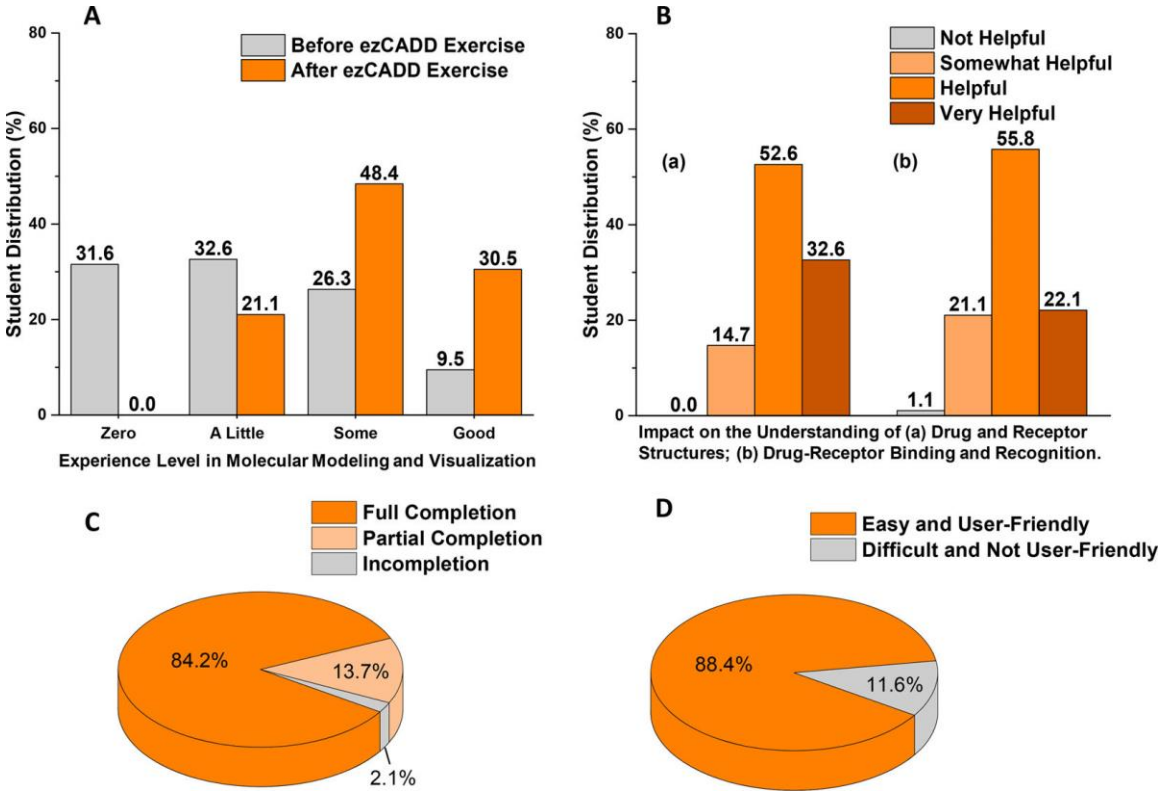


Figure 18 Student evaluation of ezCADD. (A) Student experience level in molecular modeling and visualization before and after the ezCADD exercise. (B) Impact of ezCADD on student understanding of drug-receptor structure and recognition. (C) Student completion status of the ezCADD exercise. (D) Student user experience with ezCADD.

I.8 ezHTVS: High Throughput Virtual Screening

Traditional high throughput virtual screening is based on the ligand, which usually does not require a profound understanding of the target receptor. Ligand-based high throughput virtual screening also has the advantage in speed when millions or billions of compounds need to be screened. However, this method is limited by the structure and pharmacophore of the ligand itself, which means compounds with similar structure or pharmacophore will usually have a higher ranking. The structure-based high throughput virtual screening, also known as molecular docking, requires the structure of the target receptor. Although performing molecular docking with a large compound database takes a much longer time than ligand-based high throughput virtual screening, structure-based high throughput virtual screening has relatively higher accuracy in prediction and could also get higher ranking compounds not only limited by the similarity of structure or pharmacophore.

ezHTVS combines the advantages of both structure-based and ligand-based high throughput virtual screening methods. First, ligand-based high throughput virtual screening will be performed based on the similarity between query and target structures. Then the top-ranking compounds will be selected for structure-based high throughput virtual screening, which is energy minimization with the target receptor. 13 databases are implemented in ezHTVS with more than 52 million drug-like compounds (criteria: molecular weight < 600, and -4 < logP < 6) available (detail is shown in Table 2). 11 different types 2D molecular fingerprints (Avalon, FCFP, Morgan, Pairs, Pattern, RDK, Rdmaccs, and Torsions from RDKit, and Circular, Path, and Tree from OpenEye toolkit suite) and 2 types of 3D molecular similarity search and alignment (FastROCS and

Inertial At Heavy Atoms methods from OpenEye toolkit suite) methods are provided, along with the 13 databases.

Frontend Implementation

In this application, the receptor file can be uploaded in PDB format and the ligand file can be uploaded in PDB, MOL2, or SDF/MOL format. The receptor structure could be either experimentally determined by X-ray crystallography, NMR spectroscopy, or EM, or computationally predicted by *ab initio* protein modeling or homology modeling. The ligand structure could be determined by the experimental methods or computational methods such as molecular docking. User can choose one interest database from the dropdown menu of database option, and one search method in either 2D or 3D molecular similarity search methods. One single ezHTVS task usually will be finished within one minute, with up to 100 top ranking results. 2D or 3D molecular similarity score and Vina score will be listed along with the compound ID, which is hyperlinked with the corresponding database page. A 2D molecular viewer with the 2D structure of the corresponding compound will be showed up when clicked on the item on the left side result panel, and the 3D structure in the main window will also be changed to the current selection. A zip file that contains user receptor and ligand input files, top-ranking compound structures in an SDF file, and top-ranking compound scores in a tab-delimited TSV file is also available for users to download and further analyze.

Backend Implementation

In the backend of ezHTVS, a PHP script is in control of receiving parameters from the frontend JavaScript, processing parameters and passing them to Bash and Python scripts, and then return the results to the frontend JavaScript. As shown in Figure

19 and Figure 20, all the parameters will be passed from JavaScript to PHP script by the variable "$_POST". In line 58, variable "$ext" will hold the extension of the ligand file. Database, type of molecular similarity search, and search method will be stored in variables "$database", "$twoD_threeD", and "$method" respectively in lines 63 to 65. Then the main function "fpsMain" will be called in line 67. The function "fpsMain" mainly parses the type of molecular similarity search to determine whether it's a 2D or 3D search.

If the selected search method is 2D, then the function "fpsTwoD" will be called in lines 49 and 50. In the function "fpsTwoD" (line 3 to 38), the array "$arr_fps" holds all the names of molecular fingerprints and has all the lowercase value except "RDK". Arrays "$arr_oe_fps" and "$arr_rdkit_fps" holds all the names of fingerprints in OpenEye toolkit suite and RDKit, respectively. If the user chooses OpenEye fingerprints, then lines 29 to 31 will be executed. If the RDKit fingerprints method is chosen, then lines 32 to 35 will be executed.

If the selected search method is 3D, then the function "fpsThreeD" will be called in lines 51 and 52. The function "fpsThreeD" (line 40 to 45) controls the 3D molecular similarity search for both fastROCS and Inertial At Heavy Atoms methods.

The database screening SDF ("$ligand_out") and score ("$score_out") files, and the receptor ("$receptor") and ligand ("ligand") files will be compressed to a zip file ("$zip_out") for the user to download in line 68.

Benchmark Test

A Directory of Useful Decoys (DUD) benchmark set has 40 different targets with 2,950 ligands, and each ligand has 36 decoy compounds, constructing a database of 98,266 molecules (Huang, Shoichet, & Irwin, 2006). Because of the variety and diversity of the DUD benchmark set, we used the DUD benchmark set to perform a benchmark test with different structure-based and ligand-based molecular screening methods. One type of molecular fingerprints (extended-connectivity fingerprints with Morgan algorithm, also known as Morgan fingerprints) (Rogers & Hahn, 2010), a commercial docking program Glide (Glide, 2018-4b), and two academic docking programs Vina and SMINA (Koes et al., 2013; Trott & Olson, 2010), and a 3D shape-based similarity search program FastROCS (Rush, Grant, Mosyak, & Nicholls, 2005) were used in the benchmark test.

All the ranking and scoring have been collected from the screening results of different methods. For the analysis and comparison of the accuracy of different methods, the receiver operating characteristic curve and enrichment curve was used. All the average with standard deviation and median results of ROC-AUC, top 10 enrichment, and top 100 enrichment are shown in Table 3 (detailed results shown in Table 13, Table 14, and Table 15). Figure 21 and Figure 22 show the curves of ROC-AUC and enrichment, respectively. 23M_combo, the scoring function which combines ligand-based screening score (Tanimoto score of Morgan fingerprints similarity) and structure-based screening score (Vina score), has overall better accuracy than most of the other methods.

Table 2 Implemented databases in ezHTVS.

Name of Database	**Amount of Compounds**
ChEMBL23	1727112
FDA-Approved	2162
DrugBank	8752
DrugCentral	3965
eMolecules	17074118
HMDB	41943
LINCS	41847
MolPort	6800897
NCI-2016	284176
NCI-Diversity-Set	1605
PDB	25148
SureChEMBL	14344657
ZINC15	12100459
TOTAL	**52,456,841**

```php
<?php

function fpsTwoD($f_r, $f_l, $f_db, $f_meth, $f_lo, $f_so)
{
    // Define array of name of databases
    $arr_db = array("chembl23", "dbapproved", "drugbank",
                    "drugcentral", "emolecules", "hmdb",
                    "lincs", "molport", "nci16",
                    "ncidiv", "rcsb", "surechembl",
                    "zinc15");
    // Define array of type of fingerprints
    $arr_fps = array("Avalon"=>"avalon",
                     "Circular"=>"circular",
                     "FCFP"=>"fcfp",
                     "Morgan"=>"morgan",
                     "Pairs"=>"pairs",
                     "Path"=>"path",
                     "Pattern"=>"pattern",
                     "RDK"=>"RDK",
                     "Rdmaccs"=>"rdmaccs",
                     "Torsions"=>"torsions",
                     "Tree"=>"tree");
    // Define the unique names for oe and rdkit
    $arr_oe_fps = array("Circular", "Path", "Tree");
    $arr_rdkit_fps = array("Avalon", "FCFP", "Morgan",
                           "Pairs", "Pattern", "RDK",
                           "Rdmaccs", "Torsions");
    // 2D function
    if (in_array($f_meth, $arr_oe_fps)) {
        shell_exec('/host/apps/ezCADD/HTVS/smoldock_2D_oe.sh '.$f_r.' '.$f_l.' '.$f_db.
        ' '.$arr_fps[$f_meth].' '.$f_lo.' 2>/dev/null');
        shell_exec('/host/apps/ezCADD/HTVS/smoldock_2D_score_oe.py '.$f_so.'
        2>/dev/null');
    } elseif (in_array($f_meth, $arr_rdkit_fps)) {
        shell_exec('/host/apps/ezCADD/HTVS/smoldock_2D_rdkit.py '.$f_l.' '.$f_db.' '.
        $arr_fps[$f_meth].' 2>/dev/null');
        shell_exec('/host/apps/ezCADD/HTVS/smoldock_2D_rdkit.sh '.$f_r.' '.$f_l.' '.
        $f_lo.' 2>/dev/null');
        shell_exec('/host/apps/ezCADD/HTVS/smoldock_2D_score.py '.$f_so.' 2>/dev/null');
    }
    return null;
}

function fpsThreeD($f_r, $f_l, $f_db, $f_meth, $f_lo, $f_so)
{
    shell_exec('/host/apps/ezCADD/HTVS/smoldock_3D_main.sh '.$f_r.' '.$f_l.' '.$f_db.' '
    .$f_meth.' '.$f_lo.' 2>/dev/null');
    shell_exec('/host/apps/ezCADD/HTVS/smoldock_3D_score.py '.$f_so.' 2>/dev/null');
    return null;
}

function fpsMain($fun_1, $f_r, $f_l, $f_db, $f_meth, $f_lo, $f_so)
{
```

Figure 19 Sample PHP source code (part 1 of 2) of ezHTVS.

The file location of PHP source code of ezHTVS:

/host/apps/ezCADD/php2/htvs.php

```
49      if ($fun_1 == "2D") {
50          fpsTwoD($f_r, $f_l, $f_db, $f_meth, $f_lo, $f_so);
51      } elseif ($fun_1 == "3D") {
52          fpsThreeD($f_r, $f_l, $f_db, $f_meth, $f_lo, $f_so);
53      }
54      return null;
55  }
56
57  $receptor = "Receptor.pdb";
58  $ext = pathinfo($_FILES["Ligand"]["name"], PATHINFO_EXTENSION);
59  $ligand = "Ligand.".$ext;
60  $ligand_out = "htvs_min.sdf";
61  $score_out = "htvs_score.tsv";
62  $zip_out = 'ezHTVS_result.zip';
63  $database = $_POST["Database"];
64  $twoD_threeD = $_POST["Search"];
65  $method = $_POST["Method"];
66
67  fpsMain($twoD_threeD, $receptor, $ligand, $database, $method, $ligand_out, $score_out);
68  shell_exec('/usr/bin/zip -rq '.$zip_out.' '.$receptor.' '.$ligand.' '.$ligand_out.' '.
    $score_out);
69
70  ?>
71
```

Figure 20 Sample PHP source code (part 2 of 2) of ezHTVS.

The file location of PHP source code of ezHTVS:

/host/apps/ezCADD/php2/htvs.php

Table 3 DUD benchmark test results: ROC-AUC, Top 10 enrichment, and Top 100 enrichment.

ROC-AUC							
	2D Morgan	3D FastROCS	Glide HTVS	Glide SP	Vina Dock	Smina Dock	23M Combo
Average	0.728	0.703	0.710	0.748	0.593	0.649	0.744
Median	0.747	0.736	0.724	0.751	0.593	0.638	0.774
Stdv.	0.211	0.198	0.160	0.162	0.152	0.139	0.194
Top 10 Enrichment							
Average	0.588	0.553	0.388	0.438	0.243	0.265	0.630
Median	0.600	0.650	0.350	0.500	0.200	0.200	0.650
Stdv.	0.344	0.334	0.301	0.320	0.292	0.294	0.320
Top 100 Enrichment							
Average	0.492	0.419	0.326	0.400	0.197	0.263	0.503
Median	0.444	0.364	0.290	0.400	0.126	0.179	0.456
Stdv.	0.321	0.303	0.225	0.229	0.205	0.205	0.314

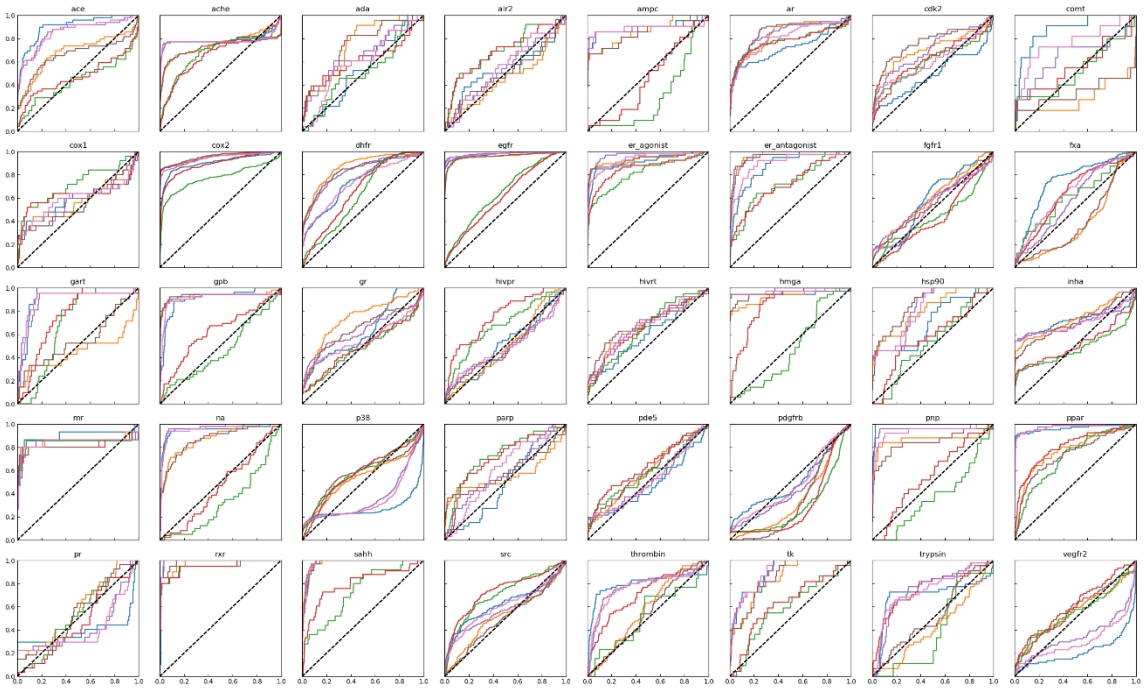


Figure 21 ROC-AUC of DUD benchmark test results. Morgan (blue), FastROCS (yellow), Vina (green), SMINA (red), 23M_combo (purple), 3M_combo (brown), and 2M_combo (pink).

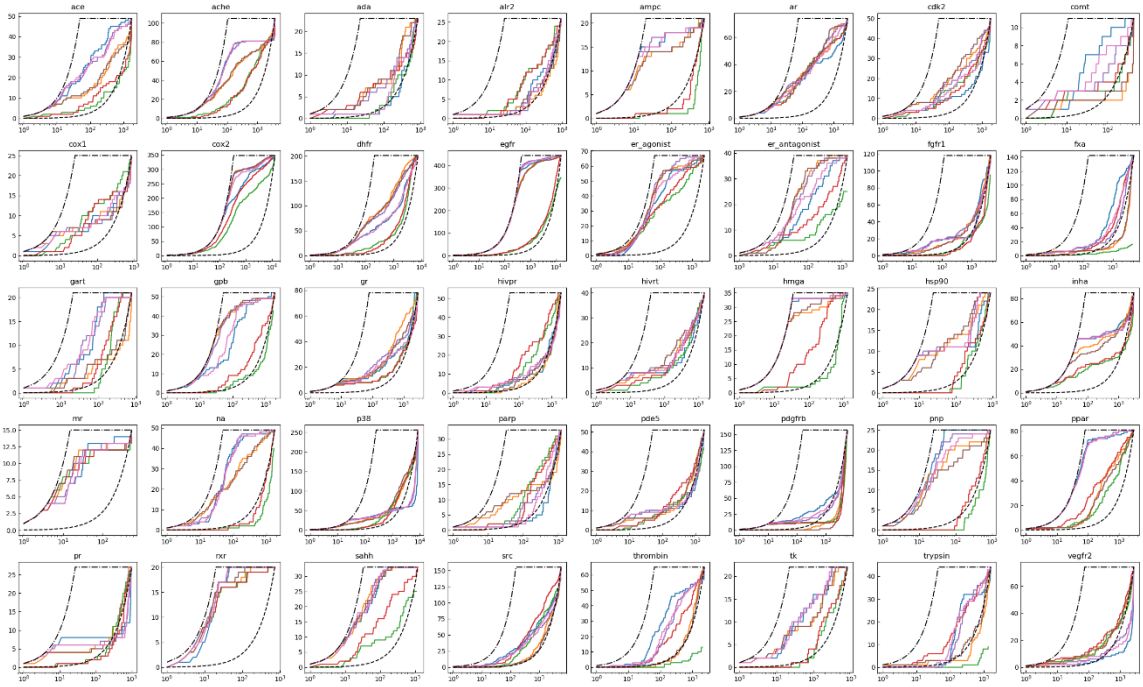


Figure 22 Enrichment of DUD benchmark test results. Morgan (blue), FastROCS (yellow), Vina (green), SMINA (red), 23M_combo (purple), 3M_combo (brown), and 2M_combo (pink).

I.9 ezPocketSearch: Drug Target and Polypharmacology Identification

In drug discovery, an ideal drug is usually designed for a specific target receptor for therapeutic effects. However, the drug molecule could potentially interact with other receptors, which often causes side effects. Although the unintended drug-target interaction is inevitable and most of the time sabotage the therapy, researchers have also been looking at the good side of multi-target interactions such as drug repurposing and multi-target treatment for different pathways (Reddy & Zhang, 2013).

ezPocketSearch is a powerful tool for multi-target search and polypharmacology identification. 340,075 protein models were collected from PDB. We then extracted all the potential pockets from the 340,075 protein models. After processing and cleaning all the pockets, a total number of 5,919,621 pockets were yielded, including 556,161 pockets with a bound ligand. Thus ezPocketSearch consists of two individual databases as the backend service, database of all pockets, and database of ligand-bound pockets.

Frontend Implementation

ezPocketSearch consists of two steps, which are controlled by the “Next” and “Back” buttons in the first window and second window of the ezPocketSearch application panel, respectively. In the first step of this application, a protein structure is needed, which could be uploaded by the user from local or provided with PDB ID in the ezPocketSearch panel. The protein structure will be processed and the second window will replace the first window in the ezPocketSearch panel after clicking the button “Next”. In the second step of this application, all the detected pockets will be shown in the main window, and the corresponding data such as ID, coordinate, and volume of pockets will be shown in the ezPocketSearch application panel for users to choose. When

the user selects a pocket in the application panel, the selected pocket will be represented in the cyan meshed surface in the main window, to be distinguished from the white meshed surfaces which are not selected. Users can choose either “All PDB Pockets” or “PDB Pockets with Known Ligands” as the target database. After clicking the “Start” button, the backend service of ezPocketSearch will be launched. One single search against the “All PDB Pockets” database will be finished within one minute, and the search against “PDB Pockets with Known Ligands” takes about 10 seconds. 100 top-ranking results will be shown on the left navigation panel, and the 100 pockets will be represented in the purple solid surface in the main window.

Case Study: Decernotinib

Decernotinib, also known as VX-509, is a Janus kinase 3 (JAK3) inhibitor in trails to study the treatment of many autoimmune diseases, for example, rheumatoid arthritis (Farmer et al., 2015). Because of its high selectivity in JAK3 inhibition, fewer side effects have been observed with decernotinib comparing with other non-selective JAK inhibitors (Fragoulis, McInnes, & Siebert, 2019). Although decernotinib has fewer side effects due to its high selectivity in JAK3 inhibition, the side effects still cannot be neglected.

A JAK3 protein with a covalent inhibitor complex (PDB ID: 5TTS) was randomly picked from PDB for the ezPocketSearch case study. First, the binding site of the 5TTS protein structure was selected after the generation of pockets in the receptor. Then the selected binding pocket was searched against the “All PDB Pockets” database. In the top 100 results, Janus kinase 3 (PDB ID: 5TTS), Janus kinase 2 (PDB ID: 4E6D), Janus kinase 1 (PDB ID: 4E4L), tyrosine kinase 2 (PDB ID: 4PYL), and others were found

(results shown in Figure 23). The binding pockets from 5TTS, 4E6D, and 4E4L were chosen and used for small molecule docking with decernotinib drug molecule in ezSMDock (results are shown in Table 4). In this docking study, decernotinib has the best docking score with the JAK3 binding pocket, and good docking scores in other JAK protein binding pockets, which confirms that decernotinib is a selective JAK3 inhibitor.

Case Study: Levonorgestrel

Levonorgestrel (LNG), for most people known as a birth control drug with the brand name Plan B, is a hormonal medication for birth control, emergency contraception, and hormone therapy. As a progestogen, although Levonorgestrel is a highly selective agonist of the progestogen receptor, it works as a very weak agonist on the androgen receptor. The hormonal effects of levonorgestrel on other target receptors can be neglected, such as estrogen, glucocorticoid, and mineralocorticoid receptors (Kuhl, 2005).

A progestogen receptor with bound levonorgestrel complex (PDB ID: 3D90) was used in ezPocketSearch. The binding site of progestogen receptor was selected in the second step of ezPocketSearch and further used in the search against the “All PDB Pockets” database. In the top 100 results, progesterone receptor (PDB ID: 3D90), glucocorticoid receptor (PDB ID: 3RY9), mineralocorticoid receptor (PDB ID: 2OAX), androgen receptor (PDB ID: 4OFR), and others were found. The binding pocket of 3D90, 3RY9, 2OAX, 4OFR, and 6CBZ (estrogen receptor) were used for small molecule docking with levonorgestrel drug molecule and endogenous ligands respectively in ezSMDock. All the docking results are shown in Table 5.

In the progestogen receptor, levonorgestrel has a docking score of -11.5, much better than -8.7, the docking score of the endogenous ligand progestogen. Thus the experimental relative binding affinity 323%, which is the binding affinity of levonorgestrel in comparison with a progestogen molecule in progestogen receptor, has been well proved in the docking results comparison. The docking results of levonorgestrel in glucocorticoid receptor, mineralocorticoid receptor, androgen receptor, and estrogen receptor are lower than the endogenous ligands respectively, which also validates the experimental relative binding affinity. The combination methods of ezPocketSearch and ezSMDock used in the case study of levonorgestrel proves that 1) the computational results are in accordance with the experimental data, 2) levonorgestrel is a highly selective agonist of progestogen receptor, and 3) levonorgestrel has very little effects on other target receptors.

Table 4 Experimental inhibition constants and predicted docking score of decernotinib.

Receptor	**ezPocketSearch Similarity (0 to 2)**	**Expt. Ki (nM) (Farmer et al., 2015)**	**ezSMDock Score (Decernotinib)**
JAK3 (5TTS)	1.26	2 ± 0.7	-8.7
JAK2 (4E6D)	0.76	13 ± 0	-8.4
JAK1 (4E4L)	0.73	11 ± 0	-8.6

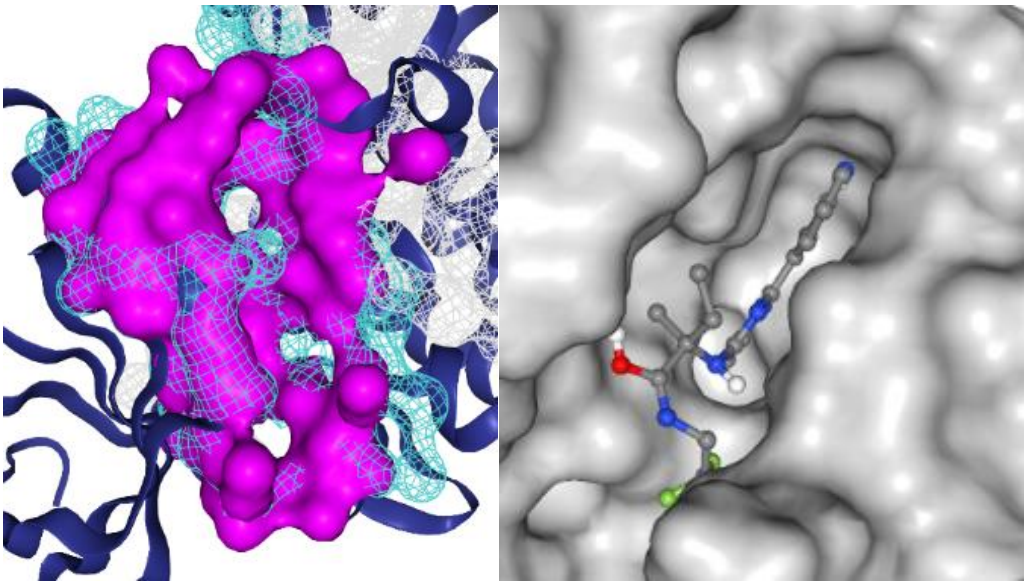


Figure 23 JAK3 (PDB: 5TTS) binding pocket search using ezPocketSearch (left), and decernotinib docked to JAK3 binding pocket (PDB: 5TTS) using ezSMDock. In the left side figure, blue meshed surface represents the query pocket from JAK3, and purple solid surface represents result pocket from JAK2 (PDB: 4E6D). In the right side figure, decernotinib drug molecule (ball and stick) docked to JAK3 (white solid surface).

Table 5 Experimental relative binding affinity and predicted docking score of levonorgestrel.

Receptor (Endogenous)	**ezPocketSearch Similarity (0 to 2)**	**Levonorgestrel Relative binding affinity (%) (Sitruk-Ware, 2005)**	**ezSMDock (Levonorgestrel)**	**ezSMDock (endogenous)**
Progesterone (Progesterone)	2.0 (3D90)	323	-11.5	-8.7
Androgen (Testosterone)	0.996 (4OFR)	58	-8.9	-10.2
Glucocorticoid (Cortisol)	1.227 (3RY9)	7.5	-10.3	-11.7
Mineralocorticoid (Aldosterone)	1.206 (2OAX)	17	-8.7	-9.5
Estrogen (Estradiol) (Milletti & Vulpetti, 2010)	Not found (6CBZ)	<0.02	-6.6	-8.1

I.10 Conclusion

ezCADD delivers a rapid, rich, smooth, dynamic, and desktop-like molecular modeling and drug design experience to biomedical researchers and students around the world through the seamless and synergistic integration of high-performance free CADD software packages with powerful web-enabled 2D/3D molecular visualization. Our ultimate goal is to break down barriers that limit access to CADD by making ezCADD a one-stop shop for the CADD needs of most researchers. The limitations of ezCADD are the same as those of the software packages used on the server-side. An advantage of ezCADD, as a web application, is the ease of software maintenance and updates on the server-side, which is completely carefree to users. ezCADD will be continuously developed and improved by the authors to address user needs and feedback. New features that will be added soon include 2D/3D protein−ligand interaction analysis and visualization, high-throughput virtual screening, fragment-based drug design, polypharmacology screening, etc. The user evaluations provided by first-year pharmacy students confirmed ezCADD's robustness and effectiveness in helping non-computational experts become self-sufficient molecular modelers and CADD practitioners. While ezCADD aims to enable biomedical discovery by democratizing CADD among traditionally underserved researchers, it also makes biomedical research more appealing to the web-orientated digital generation.

Chapter II: CADD Applications in TMC1 Study

II.1 Introduction

Hearing Loss

More than 1 billion people in the world have suffered from some degree of hearing loss. Hearing loss could happen in one or both ears in patients. Depending on the severity of hearing loss, some patients with severe hearing loss are unable to hear and having difficulties in social communication, while some patients with minor hearing loss are unaware of their condition (Global Burden of Disease Study, 2015). Hearing loss has both physiological and psychological impact on patients, leading patients with hearing loss experience more negative emotional reactions (e.g. depression, feelings of loneliness, and irritability) and social limitations than people without hearing loss (Monzani, Galeazzi, Genovese, Marrara, & Martini, 2008).

Many factors have been identified as the cause of hearing loss, for example, aging, genetic, disease, medication, noise, exposure to chemical, and physical trauma. Since there is no proven treatment for hearing loss currently available, people without hearing loss can only prevent hearing loss by avoiding the causes of hearing loss although some causes are inevitable such as aging, and people with hearing loss can only use hearing aids device or seek for alternative non-acoustic communication skills such as lip-reading and sign language.

Drug-Induced Ototoxicity

Drug (or medication) induced ototoxicity is one of the major causes of hearing loss. Patients who have been receiving treatment of aminoglycosides, glycopeptide, and

macrolide antibiotics, platinum-based anticancer drugs, loop diuretics, quinine, and salicylate analgesics could experience drug-induced ototoxicity, which is usually permanent. As the treatments are usually highly recommended due to the severe medical condition of patients, those drugs with known ototoxic side effects have to be inevitably used in patients who are in life-threatening situations. After the treatment of ototoxic drug, although the life-threatening condition might have been controlled, patients might have developed hearing loss from the drug-induced ototoxicity. Follow up costs of hundreds of thousands of US dollars were estimated for each adult who acquired permanent hearing loss due to treatment of ototoxic drugs, and the estimated follow up costs increase to 1 million US dollars for children who experienced permanent hearing loss before their language acquisition (Lanvers-Kaminsky, Zehnhoff-Dinnesen, Parfitt, & Ciarimboli, 2017).

Aminoglycoside Antibiotics Induced Ototoxicity

Aminoglycoside antibiotics are primary medications for the treatment of infections caused by gram-negative bacteria (e.g. Enterobacter, Pseudomonas, Acinetobacter, etc.). More than 10 million doses are prescribed every year in the United States. The primary mechanism of action of aminoglycosides is inhibition of protein synthesis in gram-negative bacteria (Mingeot-Leclercq, Glupczynski, & Tulkens, 1999).

In the cytosol of the bacterium, aminoglycoside antibiotic binds to the aminoacyl site (A site) of 16S ribosomal RNA of the 30S small subunit of the bacterial ribosome, negatively interfering with the binding of formyl-methionyl transfer RNA, which is an initiation transfer RNA in the bacterium, to the 30S ribosomal subunit. The improper pairing of ribosomal RNA and transfer RNA leads to a misreading of the messenger RNA

codon and further causes protein synthesis with wrong amino acids. Thus the incorrectly synthesized proteins will be misfolded, aggregated, and eventually causing the death of the bacterium (Davis, 1987; Sharma, Cukras, Rogers, Southworth, & Green, 2007). Due to the structural difference between prokaryotic ribosomes and eukaryotic ribosomes, aminoglycoside antibiotics selectively target ribosomes in bacteria without interfering with eukaryotic ribosomes in the patient.

However, aminoglycoside antibiotics have ototoxic side effects in humans and some experimental animals. In the auditory system and vestibular system, the ototoxicity of aminoglycoside antibiotics could cause the destruction of the sensory hair cells in the cochlea and vestibular labyrinth, respectively (Brummett & Fox, 1989). It has been proved that patients with mutations in mitochondrial 12S ribosomal RNA are more susceptible to ototoxicity of aminoglycosides since the mutations in the mitochondrial 12S ribosomal RNA in humans make the eukaryotic RNA more similar to the ribosomal RNA in bacteria, which is the primary target of the bactericidal activity of aminoglycoside antibiotics (Hutchin & Cortopassi, 1994; Rybak & Ramkumar, 2007; Schacht, Talaska, & Rybak, 2012). The mechanism of aminoglycoside-induced outer hair cell death is that 1) aminoglycoside enters into hair cells through mechano-electrical transducer channel, and 2) the formation of aminoglycoside-iron complex generates reactive oxygen species (ROS) and further trigger the apoptotic pathway in the hair cell (Rybak & Ramkumar, 2007). 2% to 25% of the patients receiving single aminoglycoside treatment might develop hearing loss. When receiving multiple courses of intravenous aminoglycoside antibiotics, hearing loss might occur in more than 50% of the patients (O'Sullivan et al., 2017).

Cisplatin Induced Ototoxicity

Cisplatin, *cis*-diamminedichloroplatinum (II) or *cis*-DDP, is the most commonly used chemotherapy drug in the world for treating various types of cancers (e.g. testicular cancer, ovarian cancer, cervical cancer, breast cancer, bladder cancer, lung cancer, etc.), because of its inexpensive price and effectiveness in cancer treatments.

As an anti-cancer medication for chemotherapy, cisplatin interferes with DNA replication in the nucleus of cancer cells, causing the apoptosis of the proliferative cancer cells. After administration of cisplatin through intravenous injection, the bloodstream transports cisplatin across the entire body of the patient. Cisplatin enters into cancer cells through passive diffusion, as well as active uptake through the high-affinity copper uptake protein 1 (CTR1). In the cytosol of the cancer cell, non-selective binding of cisplatin to non-DNA target proteins will cause cytotoxicity in the cancer cell. In the nucleus of cancer cells, the N7 atoms of adenine and guanine located in the major groove of the double helix are the most reactive sites of DNA, which are nucleophilic and accessible for metal binding, including platinum-based compounds. Upon binding, cisplatin reacts with DNA and form a variety of structurally different adducts, i.e. inter-strand and intra-strand cross-links. The adducts in the nucleus further block the replication and transcription, causing the death of the cancer cell (Cepeda et al., 2007; Fuertes, Castilla, Alonso, & Prez, 2003).

It has been demonstrated by many studies that cisplatin-induced cytotoxicity is also related to excessive generation of mitochondrial ROS (Choi et al., 2015). Mutations in the mitochondrial genome are also well-studied risk factors in cisplatin-induced hearing loss. In the inner ear, glutathione S-transferases (GST) proteins are crucial for

detoxification of destructive electrophiles such as cisplatin. Mutations of GST proteins in patients receiving cisplatin treatment are associated with higher susceptibility to cisplatin-induced hearing loss (Schacht et al., 2012). Genetic variants in thiopurine S-methyl transferase (TMPT), catechol-O-methyl transferase (COMT), and several other variants, including the ATP-binding cassette transporter C3 (ABCC3) are also associated with a higher risk of cisplatin-induced hearing loss (Pussegoda et al., 2013; Schacht et al., 2012). In total, the incidence of hearing loss occurs from 11% to 97%, with an average of 62% of patients receiving cisplatin treatment (Chirtes & Albu, 2014).

Approaches of Ototoxicity Prevention

Due to the prevalence and urgency of drug-induced hearing loss, many approaches have been studied from different research groups in the last couple of decades to seek treatments for drug-induced ototoxicity. While auditory hair cells and neurons in humans cannot be regenerated and the damage of auditory hair cells is irreversible, seeking oto-protective agents has become the most popular and feasible strategy for hearing loss prevention. In mitochondria, the Bcl-2 family of proteins, caspases, and p53 are the targets of cisplatin-induced ototoxicity. Since increasing of ROS generation by ototoxic drugs (e.g. aminoglycoside antibiotics and platinum-based chemotherapy drugs) can trigger the apoptotic pathway of the hair cells, antioxidants have been tested in many studies to suppress ROS generation and reduce the oxidative stress in the cochlea. For cisplatin, CTR1 and organic cation transporter 2 (OCT-2) are the major entry ports in the cochlea of mammals, but the blockage of CTR1 and OCT-2 did not show protection significantly against the killing of hair cells in the lateral line in zebrafish (Sheth, Mukherjea, Rybak, & Ramkumar, 2017). A recent study has shown inhibition of

mechano-electrical transducer (MET) channel with the oto-protective drug can protect hair cells of the lateral line in zebrafish against cisplatin-induced hair cell death (Choi et al., 2015; Sheth et al., 2017; Thomas et al., 2013).

MET Channel and TMC1 Protein

In the cochlea, the mechanoelectrical transduction is an essential step for the conversion of the mechanical movements to electrical signals. The deflections of stereocilia lead to the opening of MET channels, initializing the process of mechanoelectrical transduction in the hair cells of the cochlea. In the mammalian inner ear, both outer hair cells and inner hair cells are mechanosensory cells for the detection of sound pressure waves that are transmitted along with the auditory organ. The one row of inner hair cells is responsible for the transmission of the sound signal to the central nervous system, and the three rows of outer hair cells are responsible for the amplification of the motility of the hair bundles (Holt & Géléoc, 2017). Protocadherin 15 (PCDH15) of the tip link connects to MET channel (the schematic diagram of "stereocilium - tip link - MET channel complex" is illustrated in Figure 24 from Kurima et al.) (Kurima et al., 2015), receives the mechanical movement from the higher stereocilium of the hair bundle, and controls the opening of MET channel. MET channel is a large protein complex sitting at the membrane of the stereocilium of the hair cell bundle (the possible schematic is illustrated in Figure 25 from Corey et al.) (Corey, Akyuz, & Holt, 2019). Many membrane proteins have been demonstrated the linkage to the MET channels, such as lipoma high mobility group IC fusion partner-like 5 (LHFPL5) / tetraspan membrane protein of hair cell stereocilia (TMHS), transmembrane inner ear (TMIE), and transmembrane channel-like proteins 1 and 2 (TMC1/2), calcium

and integrin-binding family member 2 (CIB2), transmembrane O-methyltransferase (TOMT), and phosphatidylinositol 4,5-bisphosphate (PIP_2). Among the many protein components, TMC1 has been shown the major role in the whole MET channel complex (Cunningham & Muller, 2019; Pan et al., 2018). In 2018, Holt et al. and Swartz et al. both proposed mouse TMC1 protein dimer structures based on the similarity to the transmembrane member 16A (TMEM16A), which is a voltage-gated calcium-activated anion channel protein belong to the same superfamily of TMC proteins (Ballesteros, Fenollar-Ferrer, & Swartz, 2018; Pan et al., 2018). TMC1 protein has 10 transmembrane domains (shown in Figure 26), which are helices, and assembles as a homodimer in the MET complex (Corey et al., 2019; Pan et al., 2018).

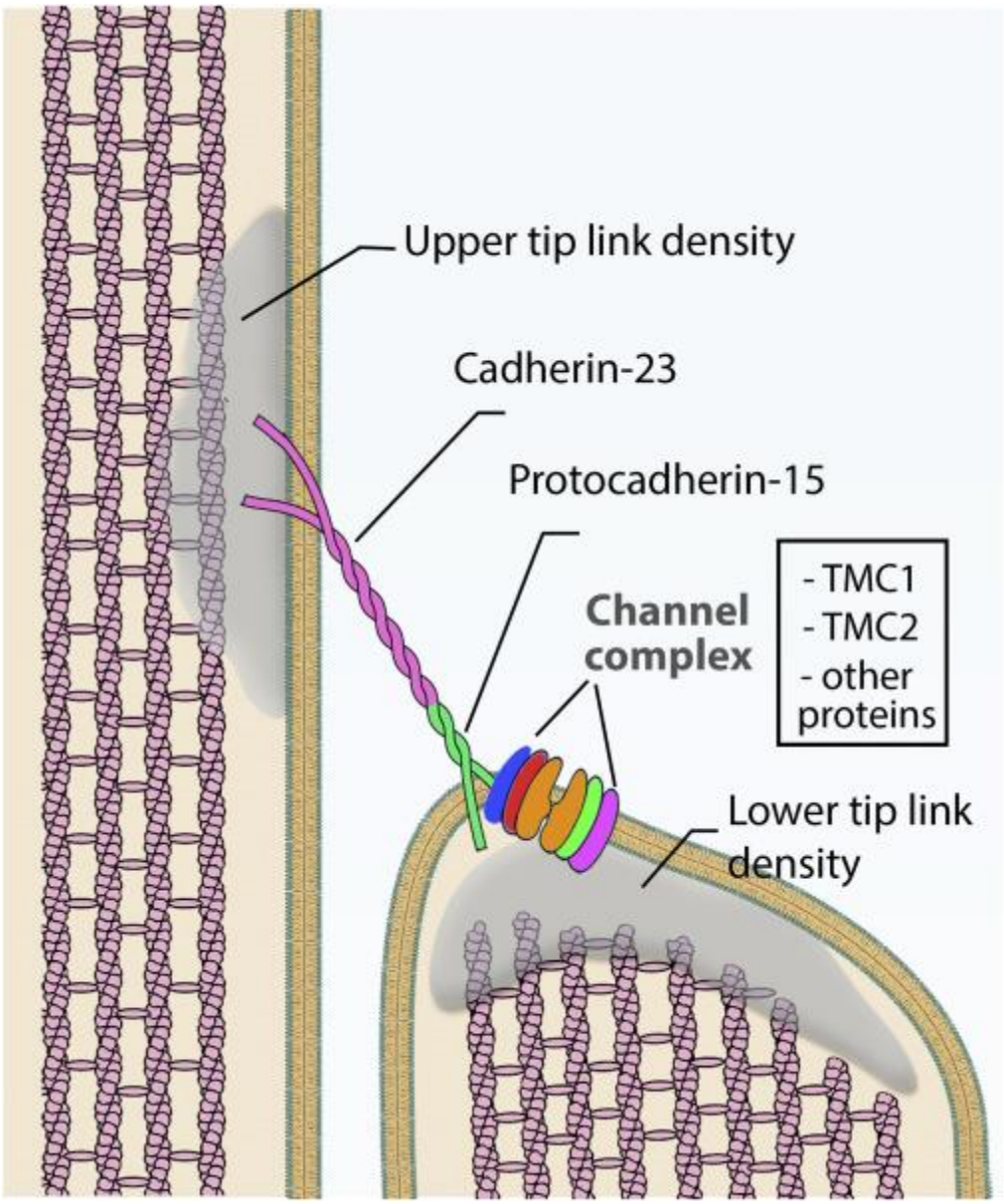


Figure 24 Schematic representation of stereocilia, tip link (cadherin-25 and protocadherin-15), and MET channel complex (from Kurima et al).

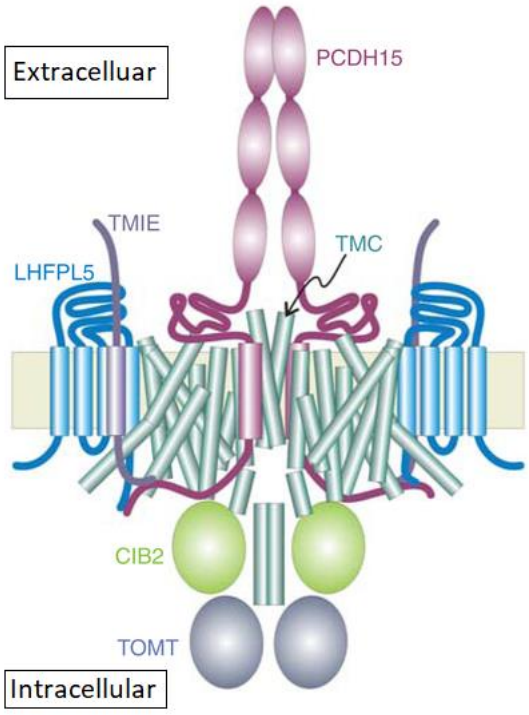


Figure 25 Schematic of MET channel complex (from Corey et al.).

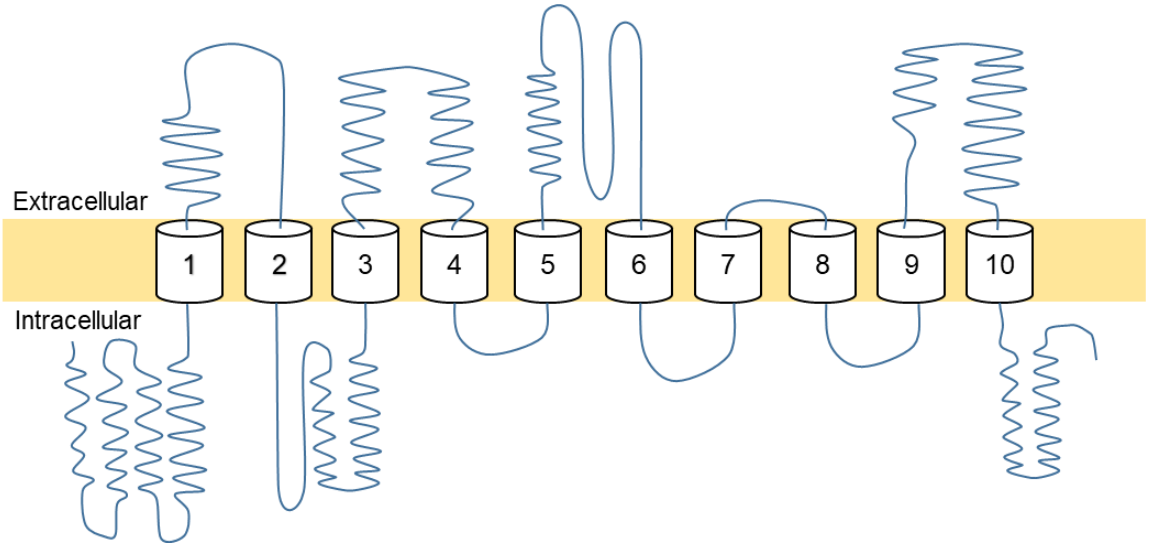


Figure 26 Predicted transmembrane topology of mouse TMC1 protein, based on the known structure of mouse TMEM16A. (modified from Holt et al.) Light yellow rectangle represents membrane.

II.2 TMC1 Homology Modelling

Human TMC1 protein has a sequence of 760 amino acids, while mouse TMC1 has 757 amino acids and zebrafish has 935 amino acids. We acquired human, mouse, and zebrafish TMC1 protein FASTA files from UniProt (Consortium, 2014) (https://www.uniprot.org/). All the FASTA files were uploaded to NCBI BLAST (Johnson et al., 2008) (Basic Local Alignment Search Tool, https://blast.ncbi.nlm.nih.gov/Blast.cgi) to perform global alignment for proteins by using the program of Needleman-Wunsch Global Align Protein Sequences (Altschul et al., 1997). Human and mouse TMC1 proteins share an identity of 95.419%. Zebrafish TMC1 protein shares around 54% identity with human and mouse TMC1 proteins (shown in Table 6).

In the workspace of the Schrodinger Maestro (Maestro, 2018-4) suite, we performed template structure alignment, multiple sequence alignment, and homology modeling.

Structure Alignment

Template structure alignment was performed based on the following detailed steps: 1) open protein preparation wizard panel, and import TMEM16A experimentally determined structure (PDB ID: 5OYB) from PDB; 2) download the mouse TMC1 homology model from the supplemental material of the paper of Swartz et al. (https://elifesciences.org/articles/38433), and then import the mouse TMC1 structure with Ca ions into Maestro user interface; 3) in the tasks menu, choose structure alignment and then choose protein structure alignment; 4) use the entire 5OYB structure as reference

structure, then superimpose the mouse TMC1 structure onto the reference structure (check the box of force alignment if needed).

Sequence Alignment

Multiple sequence alignment was performed by the following steps: 1) select mouse TMC1 structure entry and 5OYB structure entry in the project table; 2) in the tasks menu, open Prime toolkit on the applications side; 3) choose homology modeling in Prime application; 4) in the homology modeling panel, choose multiple sequence viewer; 5) in the multiple sequence viewer window, click Maestro menu button and select "Incorporate Selected Entries from Project Table" to import mouse TMC1 sequence and TMEM16A sequence; 6) click Edit menu button and select "New Sequence", then paste the zebrafish TMC1 sequence with the sequence name; 7) select all the entries in the multiple sequence viewer window, and then choose multiple alignments in the alignment menu. The result of multiple sequence alignment is shown in Figure 27.

Homology Modeling

Homology modeling was performed by the following steps: 1) in the tasks menu, open Prime toolkit on the applications side; 2) choose homology modeling in Prime application; 3) choose structure prediction wizard in the homology modeling window; 4) in the structure prediction window, paste the zebrafish TMC1 sequence as text, and then click next; 5) in the "Input Sequence" section, import the templates from project table (select mouse TMC1 structure entry and 5OYB structure entry in the project table if needed); 6) in the "Find Homologs" section, select chain A of 5OYB and chain A of mouse TMC1 structure as homologs (align selected structure if needed in the current window), and click next; 7) in the "Edit Alignment" section, run the secondary structure

prediction in the current window, and align the multiple sequences, then click next; 8) in the "Build Structure" section, choose energy-based building method, and include too Ca ions from template, then build composite/chimera model for the query sequence. The zebrafish homology model is shown in Figure 28.

The chimeric homology model from Prime has two gaps with amino acid residue number 131 to 166, and 462 to 483. The template TMEM16A structure (PDB ID: 5OYB) has four gaps (amino acid residue number 131 to 164, 260 to 266, 467 to 487, and 669 to 682) in each chain, and three of four are large gaps. In addition to the incompletion of the 5OYB template structure, the mouse TMC1 structure from Swartz et al only contains transmembrane domains that are alpha-helices, which does not include any of the extracellular or intracellular loops. Thus the major gaps of the template structures led to the incompletion of the homology model.

QUARK online server (https://zhanglab.ccmb.med.umich.edu/QUARK/) provides *ab initio* protein structure prediction and peptide folding, which constructs the 3D protein structure using amino acid sequence. In the free-modeling of CASP9 (the Critical Assessment of protein Structure Prediction) and CASP10 experiments for 3D protein structure prediction, QUARK online server was ranked No 1 of the competitions (Xu & Zhang, 2013; Yang & Zhang, 2015). We then used the QUARK online server to model the structures for amino acid sequence 128 to 169, and 459 to 486, respectively. Then we performed the superposition and fusion in Maestro user interface with the following steps: 1) import the predicted structure with amino acid 128 to 169 to Maestro; 2) open the superposition panel in the tools bar; 3) use the homology model as reference structure, choose atom pairs as the method and pick the same amino acid residues from

both structures or select the substructures with Atom Specification Language; 4) open 3D builder panel, then delete the redundant residues after the superposition; 5) choose “add bond between two selected atoms” to connect each end of the proteins; 6) choose “minimize selected atoms” to minimize the regional structure (in other edits, use “change atom properties” to change the number and name of residue if needed); 7) repeat step 1 to 6 for the predicted structure with amino acid 459 to 486; 8) open protein preparation wizard panel in tools bar; 9) under the tab of refine, in the restrained minimization section, set 0.30 Å for “converge heavy atoms to RMSD” and use OPLS3e as the force field, then perform minimization for the completed protein structure. Since the TMC1 protein complex is a homodimer, the TMC1 protein monomer was duplicated and then aligned to the chain B of mouse TMC1 structure. The chain name of the duplicate monomer was changed to B, and the chain B monomer was further merged with chain A monomer. The TMC1 dimer structure was further refined in the protein preparation wizard. The final zebrafish dimer structure is shown in Figure 29. The zebrafish TMC1 model contains two identical chains. Each chain has 12630 atoms, including 777 amino acid residues and 2 Ca ions.

Table 6 Sequence identity of human (Homo sapiens), mouse (Mus musculus), and zebrafish (Danio rerio) TMC1 protein.

TMC1 Sequence	Length	Identity (%)		
		Human	Mouse	Zebrafish
Human (Homo sapiens)	760	100	95.419	54.289
Mouse (Mus musculus)	757	95.419	100	53.139
Zebrafish (Danio rerio)	935	54.289	53.139	100

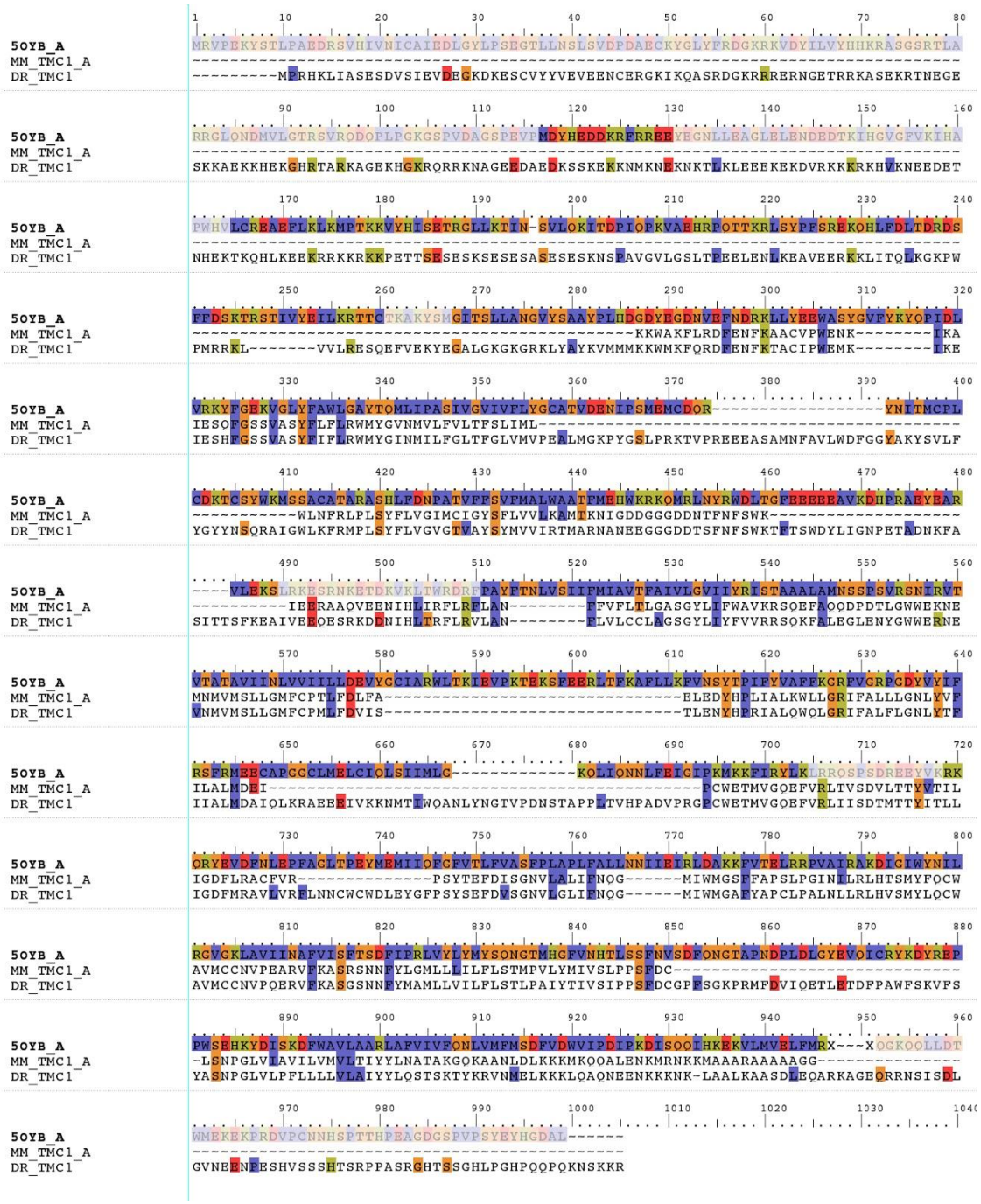


Figure 27 Multiple sequence alignment of zebrafish TMC1 (DR_TMC1), mouse TMEM16A (5OYB_A), and mouse TMC1 (MM_TMC1_A).

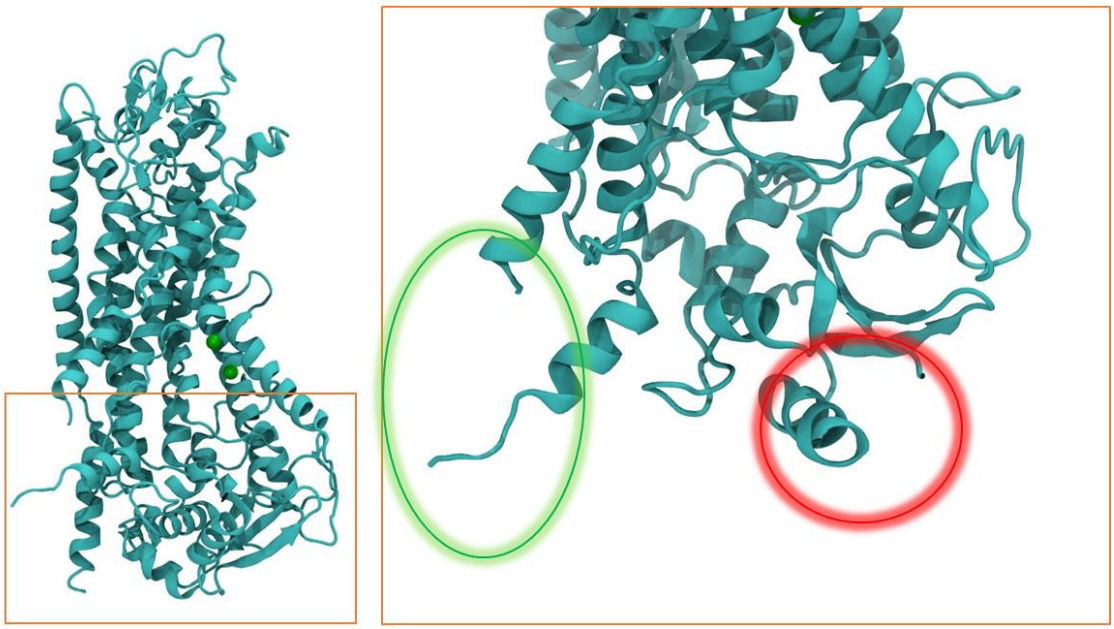


Figure 28 Homology model of zebrafish TMC1 (left). There are two gaps in the model (in the orange circle, and enlarged on the right side). Amino acid residue number 131 to 166 (in red circle), and 462 to 483 (in the green circle) are missing.

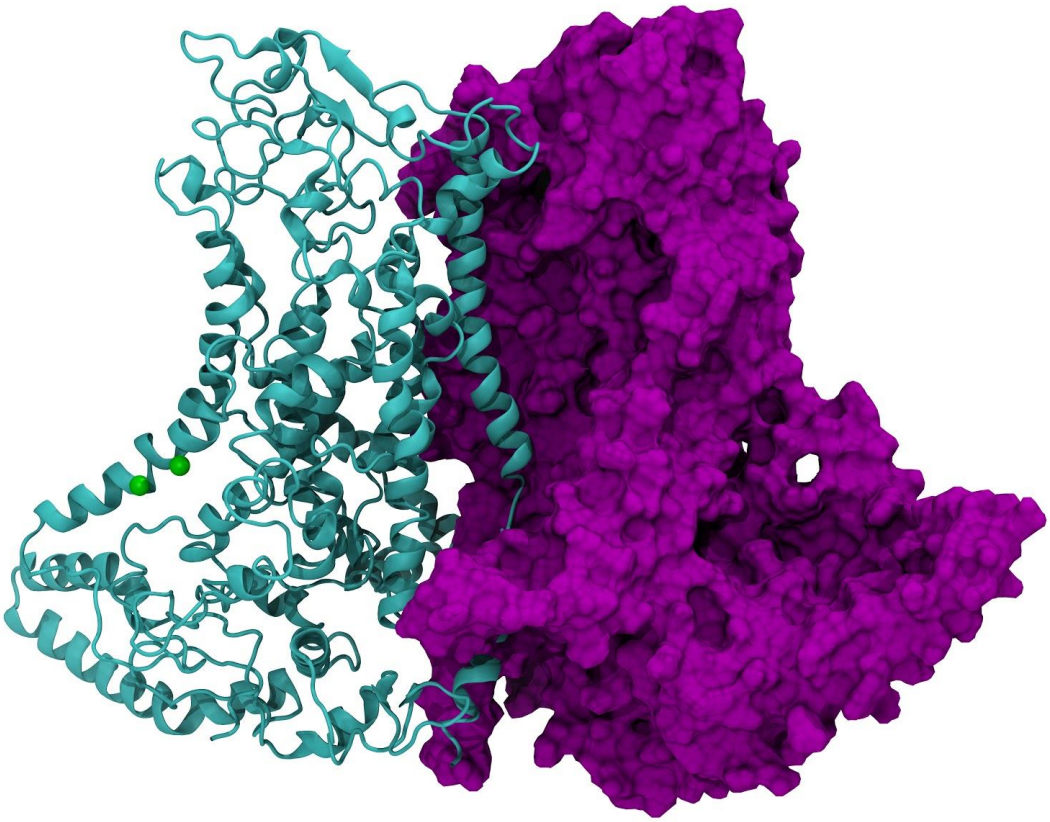


Figure 29 Zebrafish TMC1 homology model. Chain A (cyan ribbon). Chain B (purple surface). Ca ions (green sphere).

II.3 TMC1 Molecular Dynamics Simulation

Simulation System Building

In the Maestro user interface, the zebrafish TMC1 dimer structure was superimposed to the pre-aligned TMEM16A crystal structure (PDB ID: 5OYB) downloaded from the Orientations of Proteins in Membranes (OPM, https://opm.phar.umich.edu/) database (Lomize, Pogozheva, Joo, Mosberg, & Lomize, 2012). Then we performed system builder in Desmond (Desmond, 2018-4) as the following steps: 1) select the entry of zebrafish TMC1 model to present it in Maestro workspace; 2) open tasks menu, choose Desmond on the applications side; 3) in Desmond menu, choose system builder; 4) in the solvation panel of system builder window, open "Set Up Membrane" window; 5) in membrane model, choose POPC (1-palmitoyl-2-oleoyl-sn-glycero-3-phosphocholine) as the model, and then choose "Place on Prealigned Structure"; 6) the bilayer membrane will be placed based at the transmembrane domain of the TMC1 structure, use "Adjust membrane position" if needed, then click OK to exit the "Set Up Membrane" window; 7) in the solvent model, choose the predefined TIP3P water as solvent model; 8) in the boundary conditions section of the system builder window, choose orthorhombic as solvent box shape, use buffer for box size calculation method, and set 10 Å × 10 Å × 10 Å for the buffer distance (show the boundary box in the workspace, and adjust the distance if necessary); 8) in the ions panel of the system builder window, choose neutralize by adding 40 Cl ions (recalculate if needed); 9) choose add salt and set the salt concentration at 0.15 M to solvate the water system, and set K ion as salt positive ion and Cl ion as negative ion; 10) set OPLS3e (Jorgensen, Maxwell, & Tirado-Rives, 1996) (Optimized Potentials for

Liquid Simulations) as the force field, and then run the system builder job with the current setting.

The zebrafish TMC1 bilayer membrane system in the TIP3P water box has 227764 atoms, including 25260 atoms from the zebrafish TMC1 dimer protein with 2 Ca ions in each chain, 40 Cl ions for neutralization, 135 K ions and 135 extra Cl ions for solvation, 432 POPC molecules with 57888 atoms, and 48102 water molecules. The entire system was built with the OPLS3e force field.

Membrane Relaxation

Then we performed membrane relaxation for the entire protein-membrane system with the following steps: 1) save the newly built zebrafish TMC1 membrane system in a CMS file with a name *DR_TMC1_membrane.cms*; 2) use Schrodinger's python script relax_membrane.py to prepare input files with the following commands: $SCHRODINGER/run relax_membrane.py *DR_TMC1_membrane.cms* -t 300 -j *DR_TMC1_membrane-relaxation* -gpu; 3) execute the membrane relaxation protocol with the following commands: $SCHRODINGER/utilities/multisim -JOBNAME *DR_TMC1_membrane-relaxation* -HOST localhost -mode umbrella -cpu 1 *DR_TMC1_membrane-relaxation-in.cms* -m *DR_TMC1_membrane-relaxation.msj* -o *DR_TMC1_membrane-relaxation-out.cms* -set 'stage[1].set_family.md.jlaunch_opt=["-gpu"]'. The membrane relaxation can also be performed in the graphical user interface of the newer version (2018-4 or newer) of Schrodinger Desmond. When using the newer version of Maestro, check "Relax membrane model system" in the molecular dynamics panel before the simulation.

Molecular Dynamics Simulation

After the membrane relaxation protocol was finished, in Schrodinger Maestro, we loaded the *DR_TMC1_membrane-relaxation-out.cms* file into Desmond molecular dynamics panel. We set the parameters with following steps: 1) set simulation time to 100 ns, and set recording interval as 2 ps for trajectory and energy; 2) set NPT for the ensemble class; 3) set 300.0 K for temperature and 1.01325 bar for pressure; 4) check the "Relax membrane model system" if necessary; 5) open advanced options, in the integration tab, set 2.0 fs for bonded time step with 2.00 for near and 6.00 for far; 6) in ensemble tab, set Nose-Hoover chain as thermostat method, and 1.0 ps for the relaxation time; 7) in barostat section, set Martyna-Tobias-Klein as barostat method, 2.0 ps for relaxation time and isotropic as coupling style; 8) in the interaction tab, set cutoff for short range method, and 9.0 Å as cutoff radius; 9) in the miscellaneous tab, choose random seed, and check randomize velocities at the starting time of 0.0 ps; 10) in the molecular dynamics panel, choose 1 GPU for the molecular dynamics simulation; 11) set the name for the simulation and execute the 100 ns simulation job.

After the 100 ns molecular dynamics simulation, the trajectory analysis was performed with RMSD (shown in Figure 31) and RMSF (shown in Figure 32) of alpha carbon of TMC1 protein. The movement of the loops in the trajectory contributes mostly to the overall deviation and fluctuation of the atoms. No large conformational change has been observed in the 100 ns simulation.

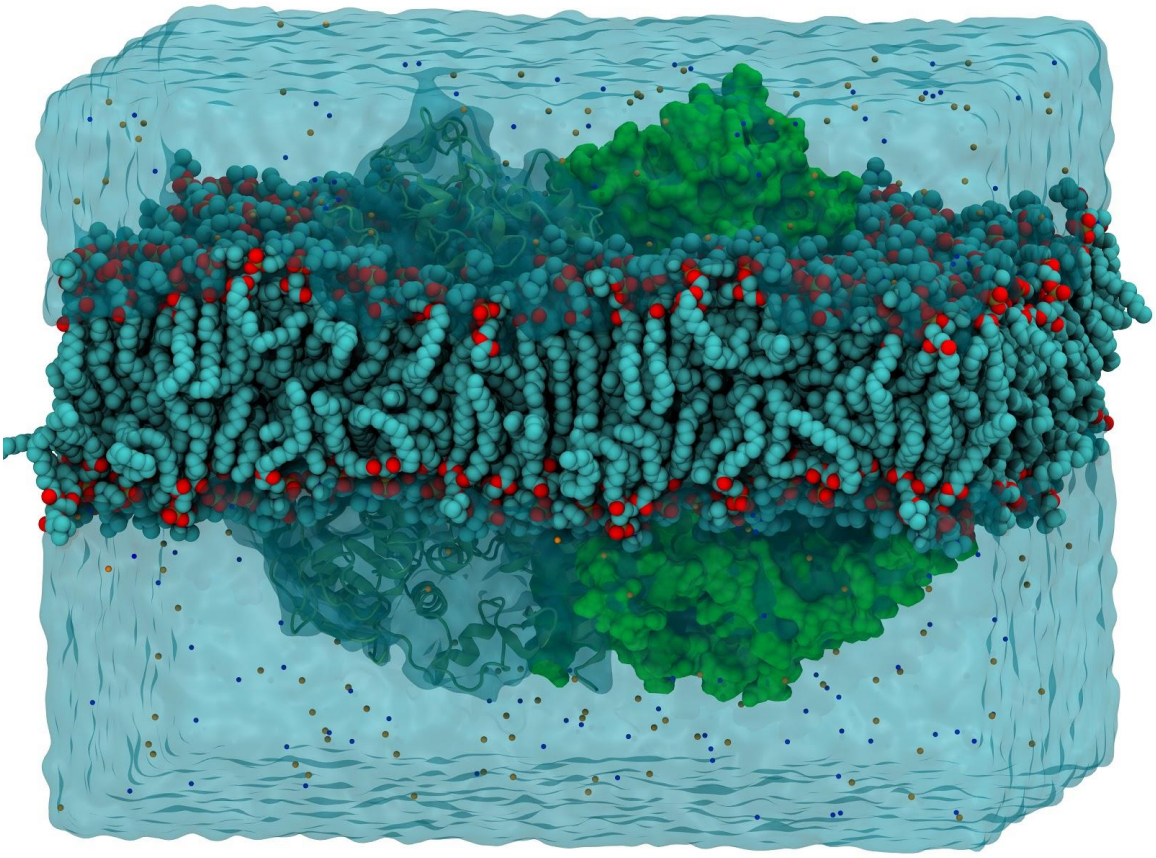


Figure 30 Zebrafish TMC1 model in bilayer membrane with explicit aqueous solvation. Chain A (green ribbon). Chain B (green surface). Bilayer membrane (VdW surface). Water box (blue). K ions (blue sphere). Cl ions (brown sphere).

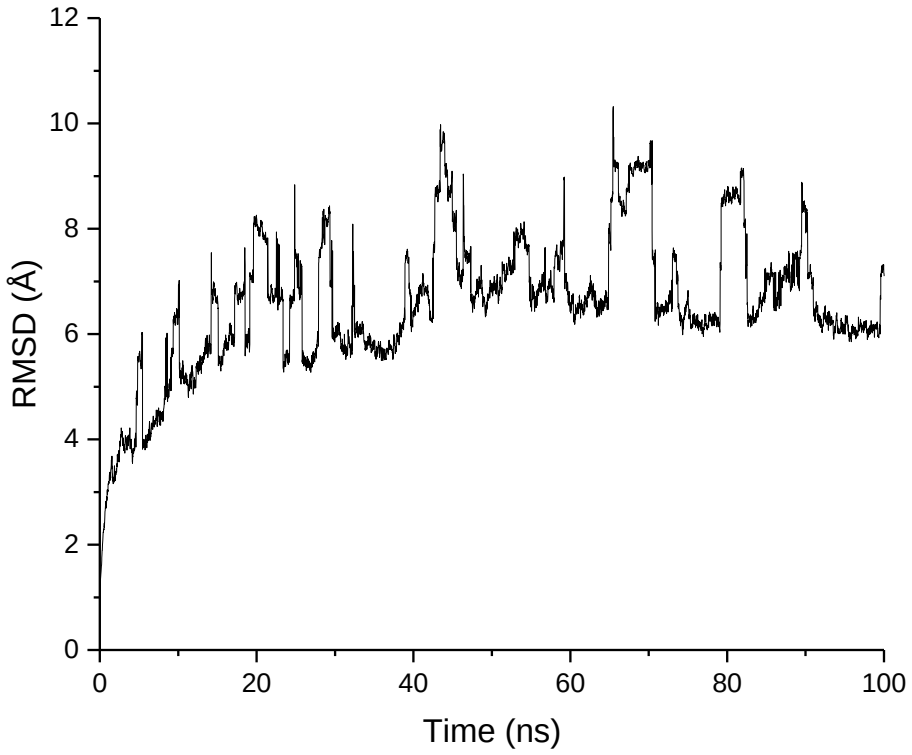


Figure 31 RMSD of alpha carbon of TMC1 protein from 100 ns MD simulation.

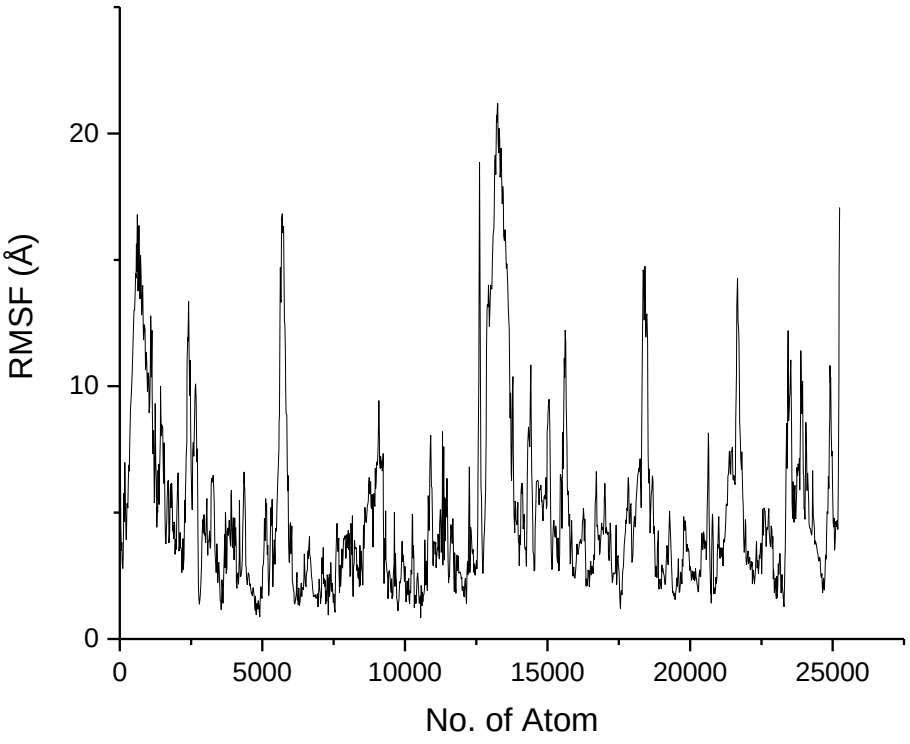


Figure 32 RMSF of alpha carbon of TMC1 protein from 100 ns MD simulation.

II.4 TMC1 Binding Site Detection

After the molecular dynamics simulation, we then looked for the binding sites of the zebrafish TMC1 model by using ezPocket web service and SiteMap (Glide, 2018-4c; Halgren, 2009), a binding site identification program from Schrodinger Maestro.

ezPocket Binding Site Detection

Three different methods (i.e., fconv, fpocket2, and fpocket3) from ezPocket were applied in binding pocket detection.

The top 2 results from fpocket2 (shown in Figure 33) are located in each chain of zebrafish TMC1 dimer. The binding pocket of chain A (with a solid surface representation) is located at the extracellular domain of chain A with the center coordinate (XYZ) of 24.5, 11.1, 20.6, and the binding pocket of chain B (with a solid surface representation) is also located at the extracellular domain with the center coordinate (XYZ) of -27.9, -10.8, 17.7. The pocket volume of chain A is around 3000 $Å^3$, and the pocket volume of chain B is around 2500 $Å^3$.

SiteMap Binding Site Detection

In Schrodinger Maestro user interface, we tried SiteMap to detect zebrafish TMC1 binding site with the following steps: 1) select the zebrafish TMC1 protein in the entry; 2) click the tasks button, then choose “Structure Analysis” in the tasks menu; 3) in “Protein Analysis” section, select “Binding Site Detection” to open SiteMap window; 4) in the SiteMap window, choose “identify top-ranked potential receptor binding sites” with all atoms in the workspace constitute the receptor; 5) in the settings section, set 15 site points per reported site for the least requirement, and set report up to 5 sites; 6) use

less restrictive definition or hydrophobicity, and use a coarse grid with the crop site maps at 4 Å from nearest site point; 7) uncheck detect shallow binding sites, and run the binding site detection job. The top predicted binding site of chain A is located in the extracellular domain of chain A with around 3000 $Å^3$ hydrophilicity volume and a total of around 6600 $Å^3$ surface volume, which is overlapped with the top predicted chain A binding site from the fpocket2 result. The top predicted binding site of chain B is located in the extracellular domain of chain B with around 3200 $Å^3$ hydrophilicity volume and a total of around 7400 $Å^3$ surface volume, which is overlapped with the top predicted chain B binding site from the fpocket2 result.

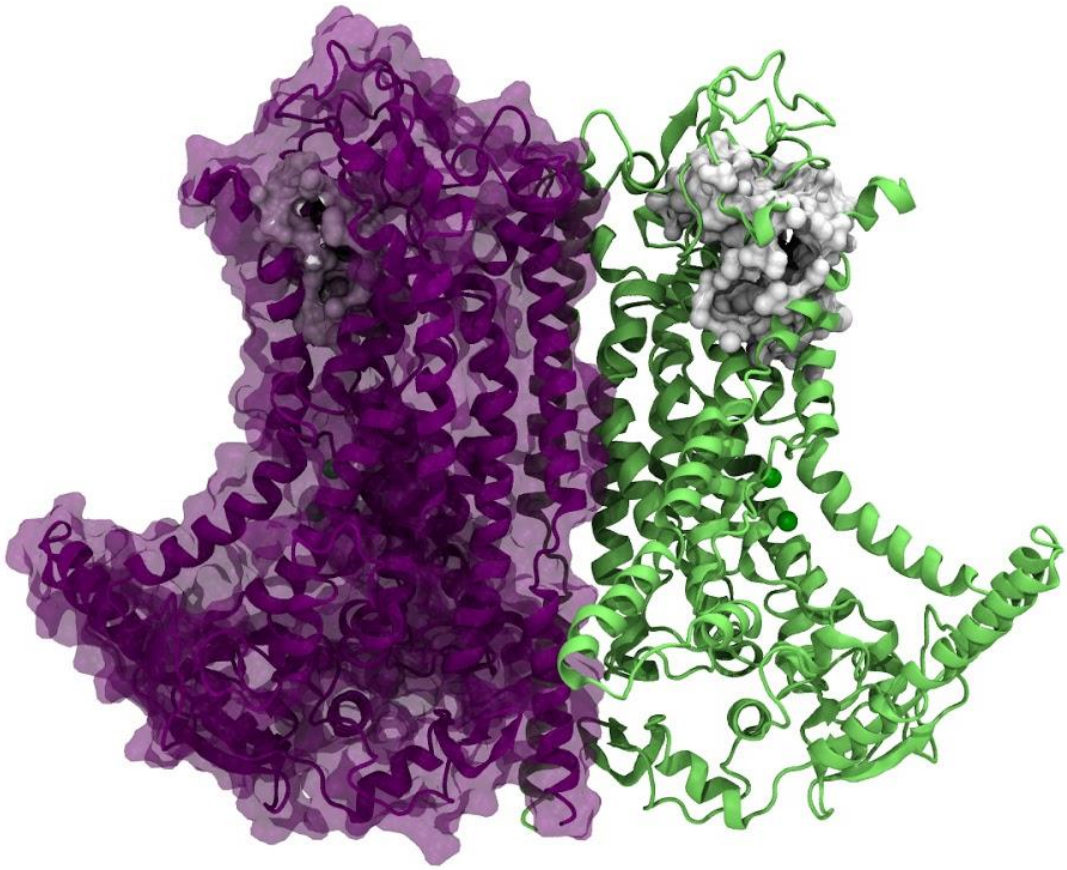


Figure 33 Potential extracellular binding sites of zebrafish TMC1 model. The center coordinate (XYZ) of binding site (in white solid surface) of chain A (in green ribbon): 24.5, 11.1, 20.6. The center coordinate (XYZ) of binding site of chain B (in purple ribbon and transparent surface): -27.9, -10.8, 17.7. Two Ca ions (green solid sphere) are in eahc chain.

II.5 High Throughput Virtual Screening

Based on the binding site prediction of ezPocket from ezCADD web service and SiteMap from Schrodinger Maestro, we performed ligand docking with small molecules, which were collected from the literature. Different docking methods were applied in small-molecule docking.

Receptor Grid Generation

Schrodinger Glide (Glide, 2018-4b) docking package was used in small molecule docking. The receptor grid generation was performed with the following procedure: 1) in the workspace of Maestro, select zebrafish TMC1 model, and then click tasks button; 2) on the application side of tasks menu, select Glide; 3) in Glide menu, select "Receptor Grid Generation"; 3) in the Receptor Grid Generation window, choose the receptor tab, and then unselect "pick to identify the ligand" box; 4) in the Van der Waals radius scaling section, set 1.0 for scaling factor and 0.25 for partial charge cutoff, and uncheck "use input partial charges" box; 5) open the advanced settings panel in the receptor tab, select OPLS3e as force field; 6) in the site tab, set 24.5, 11.1, 20.6 as the X, Y, Z coordinates of the docking center, and set 10 Å for "dock ligands with length"; 7) open the advanced settings panel in the site tab, set 20 Å for the size of midpoint box length in X, Y, and Z; 8) skip constraints, rotatable groups, and excluded volumes, then run the job for the receptor grid generation using localhost. The receptor grid will be generated to a designated zip file.

Ligand Preparation

42 compounds (shown in Table 7) have been collected from literature and purchased from different vendors, which have been shown oto-protective effect against the ototoxicity of cisplatin or aminoglycoside antibiotics in different animal studies. The simplified molecular-input line-entry system (SMILES) string and structure data format (SDF) file of each compound was collected from PubChem (Kim et al., 2019) web service (https://pubchem.ncbi.nlm.nih.gov) based on the PubChem compound identification number (CID).

After the structural data collection, ligand preparation was performed in Schrodinger with the following steps: 1) in the File menu, import the structural file that contains all the 42 compounds to Schrodinger, and then select all the 42 items; 2) in the tasks menu, choose "Ligand Preparation and Library Design", then choose LigPrep under 2D to 3D Conversion; 3) in the LigPrep window, choose the 42 selected items from the project table in the source selection; 4) select OPLS3e as force field; 5) in the ionization section, select generate possible state at target pH 7.0 ± 2.0 using Epik, and select add metal-binding states and include original state; 6) choose to desalt and generate tautomers; 7) in the stereoisomers section, choose to retain specified chiralities for computation and generate at most 32 per ligand; 8) choose SDF as output format, and perform the LigPrep task. 512 3D structures were generated from the LigPrep job of the 42 compounds.

Molecular Docking

After the receptor grid generation job was completed and all the ligands were prepared in 3D, we then performed ligand docking in Schrodinger Glide with the

following steps: 1) in tasks menu, choose Glide on the applications side, then choose Ligand Docking; 2) in Ligand Docking window, choose the receptor grid from the zip file from receptor grid generation; 3) in the ligand tab, use the ligand from SDF file or 512 selected items from the ligand preparation, and select use input partial charges; 4) set 0.80 as scaling factor and 0.15 as partial charge cutoff in the scaling of van der Waals radii section; 5) in the settings tab, choose HTVS as the docking precision; 6) select flexible for the ligand sampling method, and choose sample nitrogen inversions and select sample ring conformations; 7) choose add Epik state penalties to docking score; 8) in the output tab, choose ligand pose file (exclude receptor) in SD format for file type; 9) set at most 1 pose per ligand as the write out; 10) select perform post-docking minimization, and set 5 as the number of poses per ligand to include; 11) leave other settings as default and use all the processors in localhost for the ligand docking job, then run the current job. Repeat the ligand docking job with settings of SP (standard precision) and XP (extra precision) for the 42 compounds. The best docking result of each compound from HTVS and SP jobs are shown in Table 7 (XP data not shown).

Table 7 Experimental proven oto-protective compounds collected from the literature.

Chemical Name	**PubChem CID**	**Glide (HTVS)**	**Glide (SP)**	**Zebrafish Screen (0-2)**
Ebselen	3194	NA	NA	D
Atorvastatin calcium	60823	-2.2	-5.9	0
Omeprazole	4594	-4.1	-4.9	2
Paroxetine hydrochloride	43815	-4.9	-4.7	2
SAHA (Suberoylanilide hydroxamic acid)	5311	-0.5	-1.7	0
Quinine	1065	-5.7	-5.8	2
MR 16728 (Anti-cancer)	378811	-4.7	-5	2
EGTA	6207	-2.8	-2.8	1
Berbamine	275182	NA	NA	1
Cimetidine	2756	-2.1	-1.2	1
Bupropion	444	-5.3	-4.2	1
amifostine	2141	-4.7	-3.4	0
Trazodone	5533	-3.6	-5.5	0
Methiothepin	4106	-4	-4.8	D
Rapamycin	5284616	NA	NA	0
ML-172 (NOX1 Inhibito 3-Acetylphenothiazine)	81131	-4.2	-5.5	0
Etoposide (topoisomerase 2 inhibitor)	36462	NA	-5.5	0
CoQ10 combine Acuval 400	5281915	NA	NA	0
CoQ10 combine Acuval 400	/	NA	NA	1
Transplatin	84691	NA	NA	0
Teniposide (topoisomerase 2 inhibitor)	452548	NA	NA	0
Geldanamycin	5288382	NA	NA	D
Berbemine	10170	NA	NA	2
N-acetyl cysteine (NAC)	12035	-5.1	-5	0
Doxepin	667468	-4.3	-4.3	2
allopurinol	135401907	-4.3	-4.7	0
ML-171 (NOX1 Inhibitor 2-Acetylphenothiazine)	81131	-4.2	-5.5	0
D-methionine	84815	-3.9	-4.3	1
Sodium Thiosulfate	24477	NA	NA	0
Phenoxybenzamine	4768	-4.7	-4.5	0
Dexamethasone	5743	-4.2	-5.3	0

Table 7 Experimental proven oto-protective compounds collected from the literature (continue)

Curcumin	969516	-5.2	-6.4	0
Aspirin	2244	-4.5	-5.1	0
2-APB (2-Aminoethyl diphenylborinate)	1598	-3.5	-4.3	1
Quinine	8549	-5.7	-6	0
Edaravone (Methylphenylpyrazolone)	4021	-4.2	-5.7	0
Rasagiline mesylate	3052776	-3.6	-4	0
Rosuvastatin Calcium	446157	-4.6	-6.5	0
Pantoprazole Sodium	4679	-5.1	-5.9	0
Benzamil hydrochloride hydrate	108107	-4.4	-4.3	0
Loperamide Hydrochloride	3955	-3.5	-4.3	2
Pififthrin-u	327653	-3.7	-4.9	0

Note. Glide docking (HTVS and SP) was performed on the 42 compounds. Docking result with NA means the compound was not able to be docked in the docking site. The zebrafish screening results rank from 0 to 2. 0 means no protective effect against cisplatin. 2 means a fully protective effect against cisplatin. 1 means partial protective effect against cisplatin. D means dead in the experiment.

II.6 Machine Learning Study

235 compounds (sample data are shown in Table 10) have been collected from the literature (Hazlitt et al., 2018; Teitz et al., 2018), which were tested against cisplatin in the cell assay. The SMILES string for each compound was also collected. Mol2vec (Jaeger, Fulle, & Turk, 2018), an unsupervised machine learning technique that vectorizes the molecular structure into vector representation and based on natural language processing, was used in the vectorization of the compound dataset. 100 vectors (sample data shown in Table 11) were generated for each of the dataset compounds after the Mol2vec featurization.

Ligand Preparation

After the structural data collection and Mol2vec vectorization, ligand preparation was performed for the 235 compounds in Schrodinger with the following steps: 1) in the File menu, import the structural file that contains all the 235 compounds to Schrodinger, and then select all the 235 items; 2) in tasks menu, choose "Ligand Preparation and Library Design", then choose LigPrep under 2D to 3D Conversion; 3) in the LigPrep window, choose the 235 selected items from the project table in the source selection; 4) select OPLS3e as force field; 5) in the ionization section, select generate possible state at target pH 7.0 ± 2.0 using Epik, and select add metal-binding states and include original state; 6) choose to desalt and unselect generate tautomers; 7) in the stereoisomers section, choose to generate all combinations for computation and generate at most 1 per ligand; 8) choose SDF as output format, and perform the LigPrep task. After the ligand preparation, one 3D structure for each compound was generated.

Molecular Descriptors Generation

After the ligand preparation for the compound dataset, we generated 273 molecular descriptors with the following steps: 1) in Schrodinger user interface, select all the 235 items of 3D structure; 2) in tasks menu, choose "ADME and Molecular Properties", and then select "Molecular Descriptors" in the section of "Molecular Properties"; 3) in the Molecular Descriptors window, choose structures from project tables with 235 selected entries which are 3D structures; 4) in the Topological Descriptors tab, choose all the 226 topological descriptors; 5) in the QikProp Properties tab, choose all the 46 QikProp descriptors and unselect "Do not run QikProp with -fast option (enables PM3 calculation)"; 6) in the Semiempirical Properties tab, select Compute Semiempirical Properties and uncheck "Do not optimize geometry", and then choose AM1 as the method to use for semiempirical calculations; 7) choose CSV file as output format and run the molecular descriptors job for 235 compounds with prepared unique 3D structure. 273 molecular descriptors were generated by Schrodinger (sample data are shown in Table 12).

We then developed multiple Python scripts (sample scripts are shown in Figure 34 to Figure 37) to build different machine learning models based on SMILES string, activity data, and different features (i.e. 100 vectors generated from Mol2vec and 273 molecular descriptors from Schrodinger) of each compound.

Dataset Processing

In the dataset loading script (sample script is shown in Figure 34), we define "load_dataset" (line 1 of Figure 34) as the name of the function for loading and processing compound dataset. Four keyword arguments are defined with the default value

for this function. Keyword argument “data_type” can be “convmol” or “ecfp”. The keyword argument “fps_size” is defined with a value of 1024, which could be changed to any positive integer as needed. The keyword argument “descriptor”, with False as the default Boolean value, is defined for molecular descriptors generated by Schrodinger. The keyword argument “mol2vec”, with False as the default Boolean value, is defined for vectors generated by Mol2vec. Data files (activity data file with compound structural information: “dataset_activity.csv”; data file with the information of molecular descriptors: “dataset_descriptors.csv”; data file with the information of Mol2vec vectors: “dataset_mol2vec.csv”) are placed under the same directory of this script, as recommended. The variable “dataset_file_1” (line 3 of Figure 34) is used to hold the activity data file. CSVLoader (line 12 of Figure 34) function from DeepChem (Altae-Tran, Ramsundar, Pappu, & Pande, 2017; Subramanian, Ramsundar, Pande, & Denny, 2016) is used to load the variable “dataset_file_1” (tasks with column name “activity”, smile_field with column name “smiles”, and id_field with column name “ID”). Then the activity data with SMILES is featurized by the predefined variable “featurizer”. When using “convmol” as data type, ConvMolFeaturizer (line 8 of Figure 34) function from DeepChem will be used for generating features of graphic convolution, and the variables “descriptor” and “mol2vec” will be neglected. When using “ecfp” as data type, CircularFingerprint (line 6 of Figure 34) function will be called for fingerprints generation by using the RDKit (Landrum, 2006) Python toolkit package, with the size of the user-defined fingerprints. When the Boolean value of variable “descriptor” is True, the data file “dataset_descriptors.csv” will be loaded (line 18 of Figure 34). The first line of the data file will be read as the header line, and the first column with the title “ID” of

the data file will be read as the index. The variable “dataset” will be iterated by the attribute “id” of each object (line 19 of Figure 34). The array of molecular fingerprints and list of molecular descriptors for each compound will be concatenated and then append to the temporary list (line 20 of Figure 34). Then the shard of the dataset will be reset with the molecular descriptors (line 21 of Figure 34). By applying the same method, vectors generated from Mol2vec can be integrated into the dataset (lines 23 to 28 of Figure 34). After data integration, the dataset will be transformed. At the end of the function, variables “dataset_tasks”, “dataset”, and “transformers” will be returned (line 33 of Figure 34).

Molecular Fingerprints Generation

In the class “CircularFingerprint” (script shown in Figure 35), we define the default radius value to 2, size of binary digits to 2048, bonds value to True, features value to False, sparse value to False, and value of smiles string to False (line 3 of Figure 35). Then all the user-defined parameters will be passed to the function of “GetMorganFingerprintAsBitVect” from RDKit (line 27 of Figure 35). The array of Morgan fingerprints in the binary form will be generated and returned by the function of “GetMorganFingerprintAsBitVect”, and further returned from the current class “CircularFingerprint” (line 28 of Figure 35).

Random Forest Modeling

In the function “buildModel”, one positional argument is defined for the dataset, and two keyword arguments are defined for the fraction of training set and test set with the default value 0.6 and 0.4, respectively. The machine learning model building has 5 steps: 1) split the dataset into a training set, a validation set, and a test set based on user-

defined fractions (line 4 of Figure 36). “RandomSplitter” from DeepChem (line 3 of Figure 36) will be used to randomly split the dataset at the user-defined fractions. 2) define the ROC-AUC (area under the receiver operating characteristic curve) as the metric in classification mode (line 6 of Figure 36). 3) define “RandomForestClassifier” as the machine learning model with 100 trees in the entire forest (line 8 of Figure 36), and then train the model with the training dataset (line 10 of Figure 36). 4) after the model training, the model will be used to predict the test dataset (line 12 of Figure 36) and compute the ROC-AUC score based on the prediction data (line 13 of Figure 36). 5) the random forest machine learning model and its ROC-AUC score based on the user-defined dataset will be returned after model testing (line 15 of Figure 36).

For machine learning model building, we perform multiple trials to obtain the best and average score of the models with the following steps: 1) we will load the dataset by using the function “load_dataset” (line 1 of Figure 37); 2) define the number of trials for machine learning model building (line 3 of Figure 37); 3) define new arrays for the machine learning models and a dictionary for the best model of the trials (line 4 to 6 of Figure 37); 4) iterate the trials and get the best model with a score (line 7 to 14 of Figure 37); 5) the mean score and standard deviation of the trials will be calculated with Numpy functions (line 16 and 17 of Figure 37). To validate the machine learning method and quality of the dataset, we use the y-scrambling method to randomize the “activity” data in the dataset (the script is shown in Figure 48). One dataset will be created (line 5 of Figure 48), and the shard of the dataset will be set with randomized y data (line 12 of Figure 48). The scrambled dataset will be returned (line 14 of Figure 48) for machine learning validation.

Random forest machine learning method from RDKit was applied to train the small dataset with 235 compounds at the split ratio of 60:40 to randomly divide the dataset into a training dataset and a test dataset. The best model yielded an AUC of 0.891 with the test dataset (green curve of Figure 38). With further Y-scrambled data, the model predicted an AUC of 0.479 (red curve of Figure 38), which validated the good quality of the dataset and random forest method. Combining with the normal and Y-scrambled AUC scores, the random forest machine learning model with a small dataset shows promising performance in drug prediction.

```python
def load_dataset(data_type="convmol", fps_size=1024, descriptor=False, mol2vec=False):
    current_dir = os.getcwd()
    dataset_file_1 = os.path.join(current_dir, "dataset_activity.csv")

    if data_type == "ecfp":
        featurizer = CircularFingerprint(size=fps_size)
    elif data_type == "convmol":
        featurizer = dc.feat.ConvMolFeaturizer()

    dataset_tasks = ["activity"]
    loader = dc.data.CSVLoader(tasks=dataset_tasks, smiles_field="smiles", id_field="ID"
    , featurizer=featurizer)
    dataset = loader.featurize(dataset_file_1, shard_size=8192)

    if data_type == "ecfp":

        if descriptor:
            X_tmp = []
            t = pd.read_csv("dataset_descriptors.csv", sep=",", header=0, index_col=0)
            for x, id in enumerate(dataset.ids):
                X_tmp.append(np.concatenate([dataset.X[x],list(t.loc[id])]) )
            dataset.set_shard(0, X_tmp, dataset.y, dataset.w, dataset.ids)

        if mol2vec:
            X_tmp = []
            t = pd.read_csv("dataset_mol2vec.csv", sep=",", header=0, index_col=0)
            for x, id in enumerate(dataset.ids):
                X_tmp.append(np.concatenate([dataset.X[x],list(t.loc[id])]) )
            dataset.set_shard(0, X_tmp, dataset.y, dataset.w, dataset.ids)

    transformers = [dc.trans.BalancingTransformer(transform_w=True, dataset=dataset)]
    for transformer in transformers:
        dataset = transformer.transform(dataset)
    return dataset_tasks, dataset, transformers
```

Figure 34 Function for compound dataset loading.

The file location of Python script for dataset processing:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/O1_datasets_processing.py

```
class CircularFingerprint(Featurizer):
    name = 'circular'
    def __init__(self, radius=2, size=2048, chiral=False, bonds=True, features=False,
    sparse=False, smiles=False):
        self.radius = radius
        self.size = size
        self.chiral = chiral
        self.bonds = bonds
        self.features = features
        self.sparse = sparse
        self.smiles = smiles
    def _featurize(self, mol):
        if self.sparse:
            info = {}
            fp = rdMolDescriptors.GetMorganFingerprint(mol, self.radius, useChirality=
            self.chiral, useBondTypes=self.bonds, useFeatures=self.features, bitInfo=
            info)
            fp = fp.GetNonzeroElements()  # convert to a dict
            # generate SMILES for fragments
            if self.smiles:
                fp_smiles = {}
                for fragment_id, count in fp.items():
                    root, radius = info[fragment_id][0]
                    env = Chem.FindAtomEnvironmentOfRadiusN(mol, radius, root)
                    frag = Chem.PathToSubmol(mol, env)
                    smiles = Chem.MolToSmiles(frag)
                    fp_smiles[fragment_id] = {'smiles': smiles, 'count': count}
                fp = fp_smiles
        else:
            fp = rdMolDescriptors.GetMorganFingerprintAsBitVect(mol, self.radius, nBits=
            self.size, useChirality=self.chiral, useBondTypes=self.bonds, useFeatures=
            self.features)
        return fp
```

Figure 35 Function for molecular fingerprint featurization. Morgan fingerprint (from RDKit) is used in this featurization.

The file location of Python script for circular fingerprint featurization:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/O1_datasets_processing.py

```
def buildModel(dataset, frac_train=0.6, frac_test=0.4):
    # 1. Split the dataset to train and test sets
    splitter=dc.splits.RandomSplitter()
    trainSet, validSet, testSet=splitter.train_valid_test_split(dataset,frac_train=0.6,
    frac_valid=0, frac_test=0.4)
    # 2. Use ROC-AUC as metric
    metric = dc.metrics.Metric(dc.metrics.roc_auc_score, mode="classification")
    # 3. Model training
    sklearn_model = RandomForestClassifier(class_weight="balanced", n_estimators=100)
    model = dc.models.SklearnModel(sklearn_model)
    model.fit(trainSet)
    # 4. Model testing
    y_pred = model.predict_proba(testSet)
    score = metric.compute_metric(testSet.y, y_pred, testSet.w)
    # 5. Return score and model
    return score, model
```

Figure 36 Function for random forest classification model. RandomSplitter is used to split training and test datasets. ROC-AUC is used for the metric.

The file location of Python script for random forest classification model:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/O2_datasets_rf_class.py

```
tasks, dataset, transformers = load_dataset(data_type="ecfp", fps_size=64, descriptor=
True, mol2vec=True)

n_trials = 100
models = {}
scores = []
best_model={"score":-1, "number":-1, "model":object}
for i in range(n_trials):
    models[i] = {}
    models[i]["score"], models[i]["model"] = buildModel(dataset)
    scores.append(models[i]["score"])
    if models[i]["score"] > best_model["score"]:
        best_model["score"] = models[i]["score"]
        best_model["model"] = models[i]["model"]
        best_model["number"] = i

meanScore = np.mean(scores)
stdScore = np.std(scores)
```

Figure 37 Multiple trials of machine learning modeling and scoring (average and best). 100 trials is used in this sample script.

The file location of Python script for multiple trials of model building and scoring:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/O2_datasets_rf_class.py

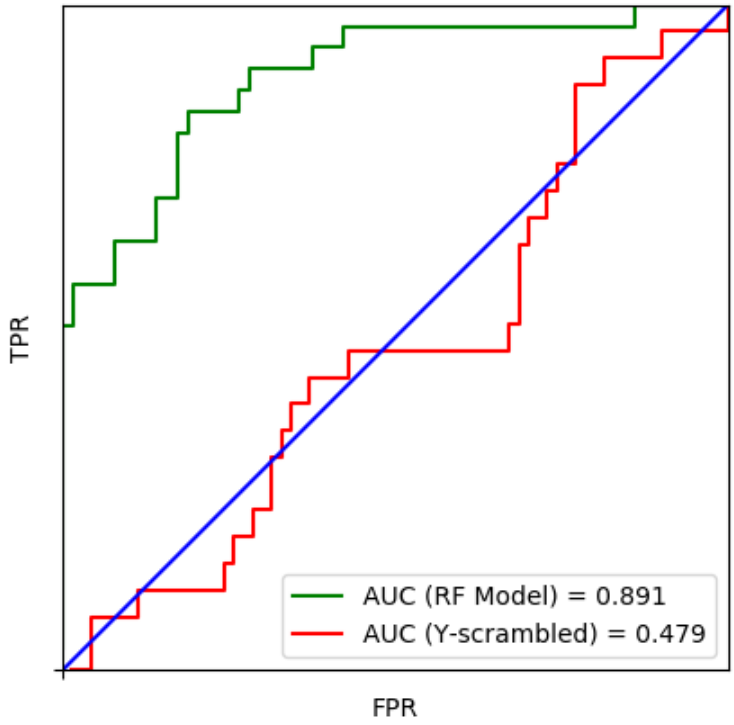


Figure 38 ROC curve of random forest machine learning model using Morgan Fingerprints. FPR: false positive rate. TPR: true positive rate. AUC: area under the curve. The green curve is from random forest model in prediction of the test set. The red curve is from Y-scrambled model. The blue line is the random line that has an AUC of 0.5.

II.7 Quantitative Structure-Activity Relationship Study

We also used the same dataset with 235 compounds (sample data are shown in Table 10) that used in machine learning study for QSAR modeling study. In Schrodinger Maestro, we used two different QSAR modeling methods to build models, which are traditional AutoQSAR (Dixon et al., 2016; Glide, 2018-4a) and AutoQSAR/DeepChem (Altae-Tran et al., 2017; Subramanian et al., 2016). AutoQSAR/DeepChem uses the convolutional neural network, a popular image processing method in machine learning, to process molecular structures.

Traditional AutoQSAR Modeling

With the CSV file of compound dataset, we perform traditional AutoQSAR with the following steps: 1) in Maestro menu, click "File" and select "Import Structures"; 2) select the target CSV file in the import window; 3) in the "Import SMILES" window, select SMILES column of the CSV file as input SMILES column and compound ID column of the CSV file as input Entry Title column, then uncheck the box of "Discard any additional properties"; 4) select all the 235 entries and group them into a new group; 5) select the entry of the created group; 6) in tasks menu, select "Discovery Informatics and QSAR"; 7) in the "Discovery Informatics and QSAR" menu, select "AutoQSAR" in QSAR group; 8) in AutoQSAR window, select "build model" as QSAR task; 9) in the "Build model" section, use structures from project table (selected entries), and uncheck "use validation set"; 7) choose "i_canvas_activity" as the prediction property, and then choose categorical as the property type; 8) in the options section, set 2 for the number of categories; 9) set 75 % for the random training set, and set 10 as the number of models to keep; 10) in the advanced options panel, set "Equal widths" for the categories, set 50 for

the numbers of models to build for each model type (MLR, PLS, etc), and set 0.80 for the maximum allowed correlation between any pair of independent variables; 11) in the descriptors section of advanced options, select “Binary fingerprints” (radial, linear, dendritic molprint2D) and “Molecular properties” (canvasMolDescriptors), and then save the parameters; 12) in the job setting panel, choose 8 processors from localhost, and then start the AutoQSAR job.

After the AutoQSAR model building job finished, we will be able to see the details of the model and use the model to make prediction with the following steps: 1) choose “view model and make prediction” from the QZIP file under the task folder generated by AutoQSAR job; 2) the top 10 models can be seen in the model report, and all the details can be viewed by clicking the button with a plus symbol; 3) in model prediction section, select compound(s) in project table or choose to use structure(s) from file; 4) choose one method from three different modeling methods to test, which are all models (consensus prediction), best model, and selected models (consensus prediction); 5) type the name for AutoQSAR prediction, and then launch the prediction job. After the AutoQSAR prediction task, a new entry of the predicted compounds will be generated in the entry list. We can check the prediction result in the corresponding column with the customized task name in the project table.

The best model has a ranking score of 0.7518. The confusion matrix of the test dataset was also generated after modeling. In the confusion matrix, 36 are true positive, 15 are true negative, 3 are false positive, and 4 are false negative (see Table 8) for the confusion matrix). We then calculated sensitivity, specificity, precision, negative predictive value, accuracy, and Matthews correlation coefficient based on Equation 1,

Equation 2, Equation 3, Equation 4, Equation 5, and Equation 6, respectively. All the evaluation results of the confusion matrix are shown in Table 9.

AutoQSAR/DeepChem Modeling

With the same CSV file of the compound dataset, we perform AutoQSAR/DeepChem with the following steps: 1) in Maestro menu, click “File” and select “Import Structures”; 2) select the target CSV file in the import window; 3) in the “Import SMILES” window, select SMILES column of the CSV file as input SMILES column and compound ID column of the CSV file as input Entry Title column, then uncheck the box of “Discard any additional properties”; 4) select all the 235 entries and group them into a new group; 5) select the entry of the created group; 6) in tasks menu, select “Discovery Informatics and QSAR”; 7) in the “Discovery Informatics and QSAR” menu, select “DeepChem/AutoQSAR” in QSAR group; 8) in the “AutoQSAR/DeepChem” panel, choose “Build model” as the task; 9) in the options section, choose “Classification” as the model type; 10) use structure from the selected entries in the project table, and then select “Activity” as the prediction property without adding extra descriptors; 11) use random split at 75 % to split the dataset into training set and test set; 12) set 1 hour for the model training time; 13) set 1 GPU on localhost for the current task and launch the AutoQSAR/DeepChem job.

After the AutoQSAR/DeepChem model building job finished, we will be able to check the details of the model and use the model to make a prediction with the following steps: 1) in the “AutoQSAR/DeepChem” panel, choose the load model as the task; 2) select the generated QZIP file with the customized job name for AutoQSAR/DeepChem model building; 3) select compound(s) in the entry list; 4) in the “Model Predictions”

section, choose the selected entries from project table; 5) set the output property name, and then set the job name and launch the task with 8 processors on localhost. In the "Model Summary" section, we will be able to view the full report for the current model by clicking the button of "View Full Report", and we can visualize the ROC curve (shown in Figure 39) and ROC-AUC score of the current model in the plot panel of the AutoQSAR/DeepChem report viewer window.

Table 8 Confusion matrix for the test dataset.

		True condition	
		Positive	Negative
Predicted condition	Positive	36	3
	Negative	4	15

Note. 58 compounds were used for the test of the QSAR model. 36 are true positive (TP) and 15 are true negative (TN). 3 are false positive (FP) and 4 are false negative (FN).

Table 9 Measures for test dataset confusion matrix from the best AutoQSAR model.

Measure	**Value**
Sensitivity	0.9000
Specificity	0.8333
Precision	0.9231
NPV	0.7895
ACC	0.8793
MCC	0.7229

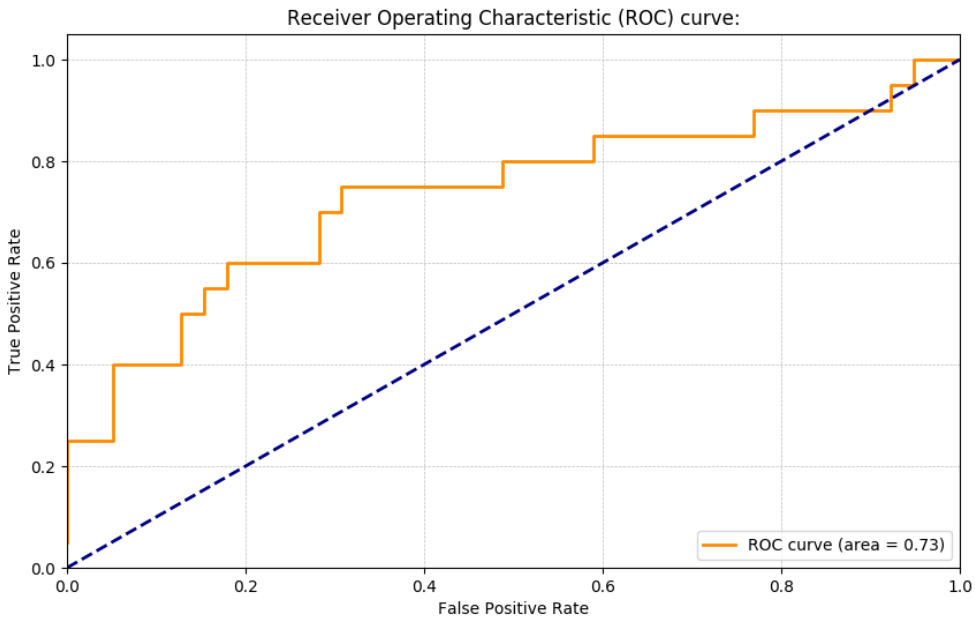


Figure 39 ROC curve for AutoQSAR/DeepChem model.

II.8 Conclusion

Drug-induced ototoxicity with the zebrafish model enables rapid high throughput screening in the studies of the ototoxic effect of a drug and the discovery of oto-protective drugs against the drug-induced ototoxicity (Ton & Parng, 2005). In this study, we built the very first zebrafish TMC1 homology model, embedded in a bilayer membrane, and performed molecular dynamic simulation. With the TMC1 protein model, we were able to perform structure-based virtual screening. Compound CID 1065 (quinine) got the best score in the Glide HTVS docking study, and also proved by a full oto-protective effect in zebrafish experiment with a score of 2. With all the compound information collected from literature, we were able to build a library for ligand-based drug design. Different techniques of machine learning and QSAR were applied to build machine learning models and QSAR models, respectively. Those models can be used in the ligand-based virtual screening with databases and improved with more experimental data. More CADD techniques can be applied in the zebrafish ototoxicity studies with different targets. With all the advantages of zebrafish screening, the results from CADD studies can be easily translated to zebrafish study.

Chapter III: CADD Applications in Mcs1 Study

III.1 Introduction

Clostridium sordellii, a Gram-positive anaerobic bacterium, can be found in soil or human intestines. *C. sordellii* can cause rapidly progressive myonecrosis with a fulminant shock syndrome especially in obstetric and gynecologic patients, which is life-threatening in many cases (Kimura et al., 2004; Ramirez & Abel-Santos, 2010). A metalloproteinase of *C. sordellii* (Mcs1) cleavage activity in humans contributes to the *C. sordellii* infections and also exhibits immune-inflammatory response and a marked leukemoid reaction (LR) (M. J. Aldape et al., 2017).

Vascular cell adhesion molecule 1 (VCAM-1) is a protein that regulates the leukocyte-endothelial cell adhesion and signal transduction (M I Cybulsky et al., 1991). Proteolytic cleavage of VCAM-1 will release the soluble ectodomain, further causing inflammatory symptoms associated with many diseases. Previous experimental studies show Mcs1 cleaves VCAM-1 protein *in vitro*, and the cleavage plays an important role in *C. sordellii* infections (Michael John Aldape, Bryant, Ma, & Stevens, 2007; M. J. Aldape et al., 2017; Myron I Cybulsky et al., 2001).

The function of Mcs1 is crucial in *C. sordellii* infections as well as in the research of disease prevention and treatment. However, since it was recently discovered, currently there is no experimentally determined 3D structure of Mcs1 or the full-length VCAM-1 structure. In this study, we used the homology modeling method to build Mcs1 and the full-length VCAM-1 homology model and performed molecular dynamics simulation for

both models. We also investigated protein-protein interaction between Mcs1 and VCAM-1 and molecular docking with the Mcs1 structure.

III.2 Mcs1 Homology Modelling

The ORF_00259 (referred to as *mcs1*) contained 1539 base pairs that encoded for a 512-amino acid protein. No identical sequence was found from GenBank (Benson et al., 2012) for the full-length *mcs1* sequence. Pfam (Bateman et al., 2004) (https://pfam.xfam.org/) and NCBI protein-BLAST (Altschul et al., 1997; Johnson et al., 2008) analyses indicated that Mcs1 has 3 domains: a fungalysin/thermolysin propeptide (FTP) domain, a peptidase propeptide, and YPEB (PEPSY) domain, and an M4 peptidase domain. No homologous structures of the FTP and PEPSY domains were found in the Protein Data Bank. Protein-BLAST searches against NCBI non-redundant protein PDB databases demonstrated that homology with Mcs1 was largely restricted to the catalytic M4 peptidase domain of the proteins. Bacillolysin (47% identity, 66% similarity) and elastase from *Staphylococcus epidermidis* (47% identity, 64% similarity), and aureolysin from *Staphylococcus aureus* (52% identify, 65% similarity) possessed the highest homology, suggesting that Mcs1 belongs to the M4 family of neutral peptidases (thermolysin-like metalloendopeptidases), a member of the zinc-dependent "GluZincin" superfamily. No Mcs1 homologs were identified in the *C. difficile* genome.

When the Mcs1 protein sequence corresponding to the catalytic M4 peptidase domain was used, the NCBI protein-BLAST search against the PDB yielded the top-ranked crystal structure (PDB ID: 1BQB) (Banbula et al., 1998) with 99 % coverage, 4×10^{-92} E-values, and 52 % sequence identity. This high-resolution structure (1.72 Å) of *S. aureus* aureolysin (referred to as aureolysin) was selected as a template for homology modeling of the Mcs1 catalytic M4 peptidase domain. Amino acid sequence alignment between the catalytic domains of Mcs1 and aureolysin showed that, similar to aureolysin,

the catalytic domain of Mcs1 is a single-chain enzyme consisting of 302 residues (Figure 40). This high degree of similarity supported Mcs1's connection to the bacterial thermolysin family of metalloproteinases.

40 candidate structures were generated by Modeller (Webb & Sali, 2014) and one candidate structure by PRIME. The energetic fitness of the 41 candidate structures, along with the template structure 1BQB as a reference, was evaluated using the DOPE score. The PRIME generated structure was considered the best among all candidates and the Modeller structure #21 the second most suitable (Figure 41 A). DOPE (Shen & Sali, 2006) per-residue based scores confirmed that the PRIME structure was energetically more favorable than Modeller structure #21 (Figure 41 B). Specifically, less per-residue deviation from template structure 1BQB was observed in the PRIME structure when compared to Modeller structure #21. The structural quality of the PRIME structure against the template structure was also evaluated using Ramachandran plots (Figure 41 C and D). Both the PRIME and template structure each had 100% of their residues within the allowed regions, with 81.4% and 87.9%, respectively, observed in the most favored region (Figure 41 C and D). Because of its comparable energetic and structural quality to the crystal structure template, the PRIME structure represented the best 3D atomic model for characterizing the functional peptidase domain of Mcs1.

Visual inspection of the Mcs1 model structure showed that the N-terminal subdomain (residues 1-156) consisted of mostly β-sheets and two α-helices. By comparison, the C-terminal subdomain (residues 157-302) contained primarily α-helices (Figure 49 A). The Mcs1 active site was located in a deep and narrow cleft between the N-terminal and the C-terminal subdomains. The entrance to this cleft was gated by loop

structures found on both the N-terminal (loop-N) and C-terminal subdomains (loop-C) (Figure 49 A). The Mcs1 loop-C is longer than its counterpart in aureolysin due to insertions. The presence of one zinc and three calcium ions were located within the active site of Mcs1 and are essential to maintaining enzymatic activity. Additionally, the residues involved in coordinating the active site (i.e., zinc ion, substrate binding, and catalysis) are conserved between Mcs1 and aureolysin (Figure 40). The calcium ions located near the active site are a common feature shared by all members of the thermolysin and neutral protease family (Figure 49 A).

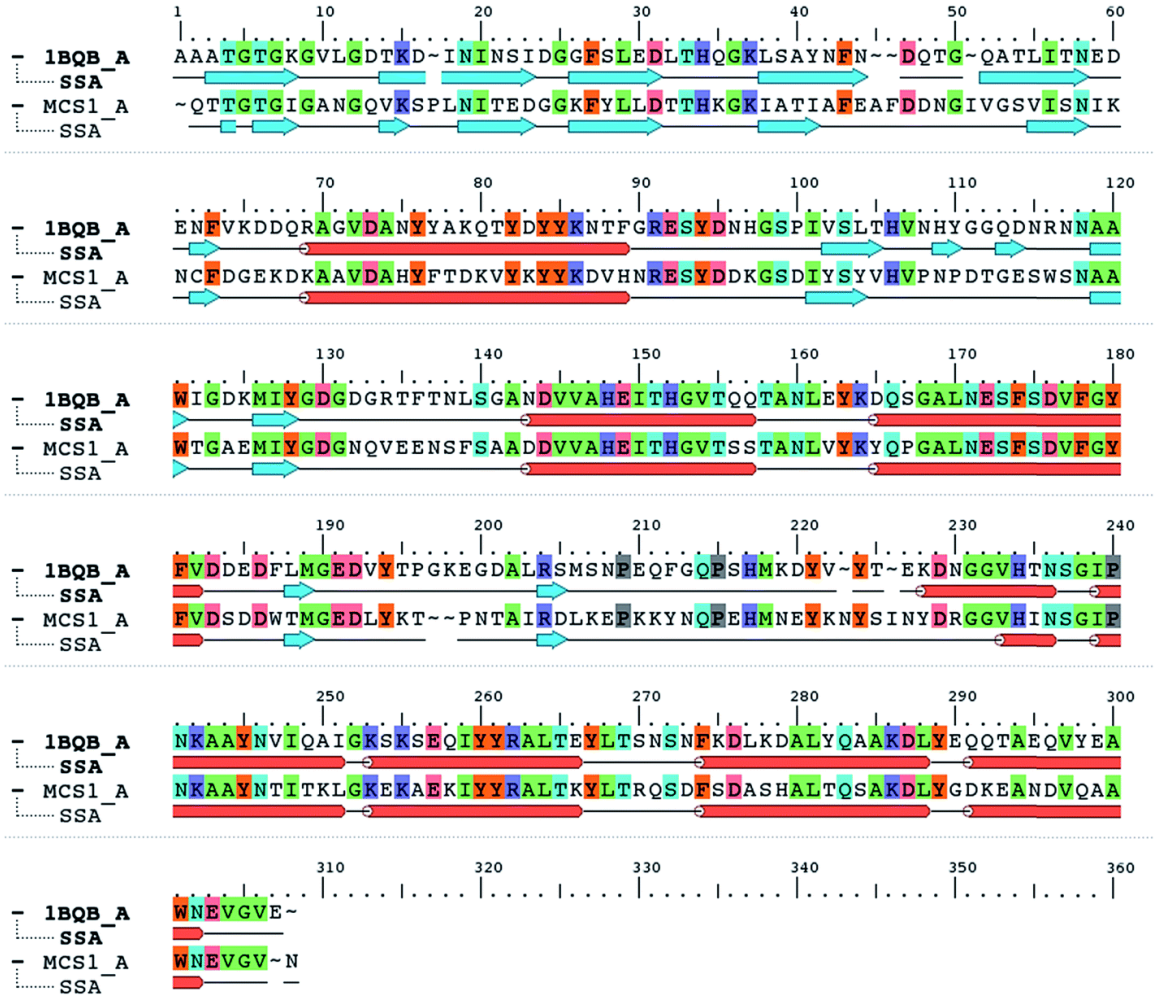
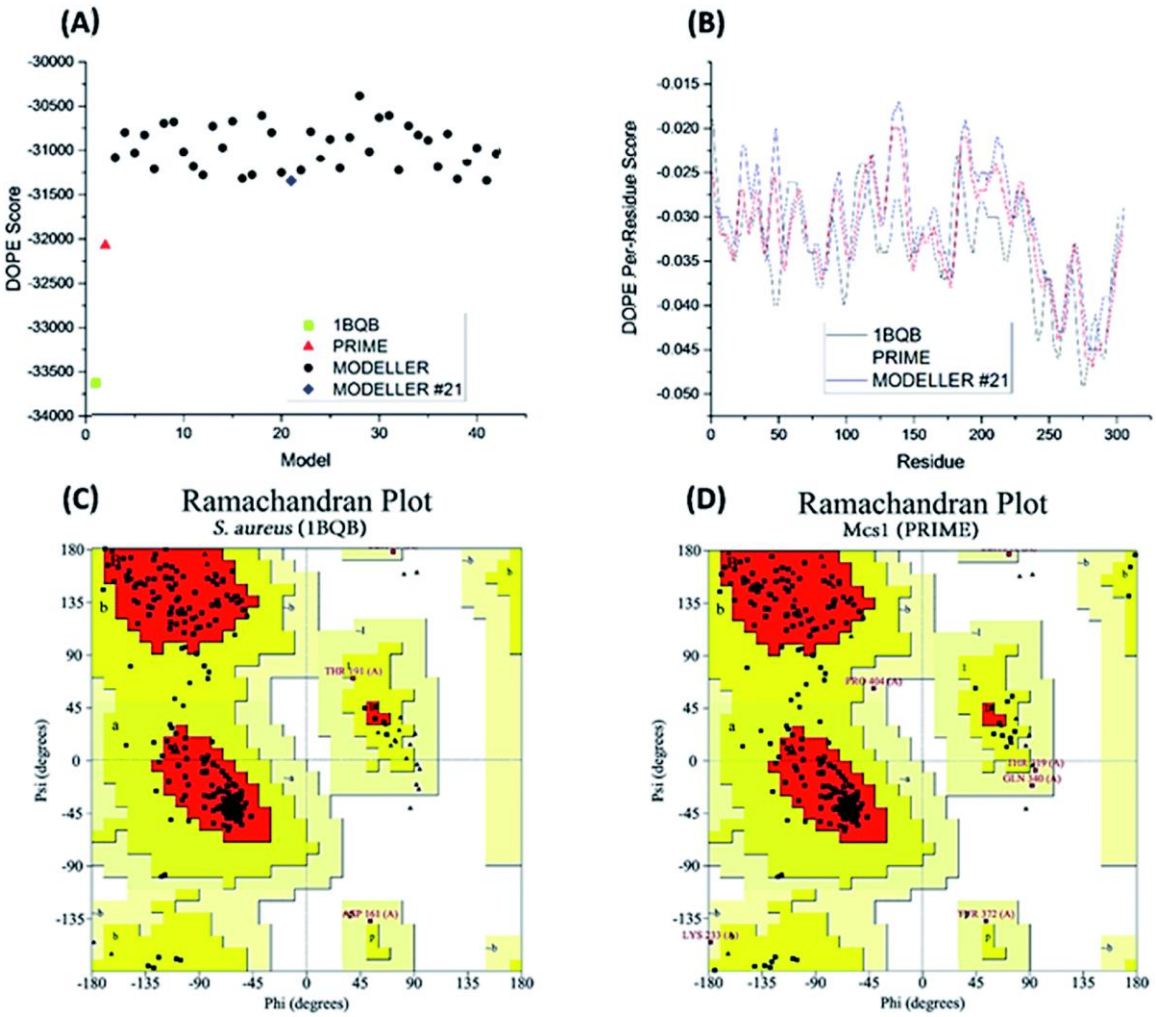


Figure 40 Protein sequence alignment of Staphylococcus aureus aureolysin (PDB: 1BQB) and C. sordellii Mcs1 catalytic domain. Conserved residues are highlighted in color. Cylinders represent α-helices and arrows represent β-strands.



Protein Structural Quality Check	*S. aureus* (1BQB)	Mcs1 (PRIME)
Residues in most favored regions	87.9%	81.4%
Residues in disallowed regions	0	0

Figure 41 Energetic and structural quality of Mcs1 model structures and the S. aureus aureolysin crystal structure template (PDB: 1BQB) as a control. (A) Overall DOPE score (lower is better); (B) DOPE per-reside scores; (C) Ramachandran plot of 1BQB; and (D) Ramachandran plot of the best Mcs1 structure modeled using PRIME.

III.3 Mcs1 Molecular Dynamics Simulation

To further optimize the best-scored structure and assess the structural characteristics of Mcs1, a 250 ns all-atom explicit solvent MD simulation was performed. Protonation states of the Mc1 protein residues were determined at pH 7 with PROPKA (Li, Robertson, & Jensen, 2005). The protonated Mcs1 structure was solvated in an orthorhombic TIP3P water box, leaving 10 Å between the solute surface and the box boundary. The Mcs1 protein and metal ions were parameterized with the OPLS 2005 force field (Banks et al., 2005). After the system was neutralized, an ionic concentration of 0.15 M NaCl was introduced to mimic experimental assay conditions. DESMOND 4.3 module of Schrodinger suite 2015-3 package (Schrödinger, 2015) was used to carry out the MD simulation. Before the production run, a 5-step simulation protocol was performed to relax the system: (1) 100 ps NVT ensemble with Brownian dynamics at 10 K with solute non-hydrogen atoms restrained; (2) 12 ps NVT ensemble using a 10 K Berendsen thermostat with a fast temperature relaxation constant, velocity resampling every 1 ps, and non-hydrogen solute atoms restrained; (3) 12 ps NPT ensemble using a 10 K Berendsen thermostat and a 1 atm Berendsen barostat with a fast temperature relaxation constant, a slow pressure relaxation constant, velocity resampling every 1 ps, and non-hydrogen solute atoms restrained; (4) 12 ps NPT ensemble using a 300 K Berendsen thermostat and a 1 atm Berendsen barostat with a fast temperature relaxation constant, a slow pressure relaxation constant, velocity resampling every 1 ps, and non-hydrogen solute atoms restrained; (5) 24 ps NPT ensemble using a 300 K Berendsen thermostat and a 1 atm Berendsen barostat with a fast temperature relaxation constant, a normal pressure relaxation constant. Finally, the 250 ns production run was performed in

the NPT ensemble without restraint. 300 K temperature and 1 atm pressure were maintained with Nosé-Hoover chain dynamics and Martyna-Tobias-Klein barostat (Martyna, Klein, & Tuckerman, 1992). A 2 fs time step and periodic boundary conditions were applied in conjunction with particle-mesh Ewald (Essmann et al., 1995) to treat long-range electrostatics. A multiple-time-stepping algorithm was employed, in which bonded interactions short-range nonbonded interactions were evaluated at every time step and long-range electrostatic interactions were evaluated every three time-steps at a cutoff of 9 Å.

Similar to the aureolysin template structure, the Mcs1 active site cleft possesses a predominantly "closed" conformation. Cleft "opening" prompts Mcs1 enzymatic activity by allowing substrates to make contact with the zinc active site located at the bottom of the cleft. Our 250 nanosecond (ns) MD simulation demonstrated that cleft opening and closing is mostly dictated by the polar interactions between Glu113 and Arg227, located on the loop-N and loop-C subdomains, respectively, on opposing sides of the cleft. Maximal "open" and "closed" Mcs1 conformations illustrate how the accessibility of the zinc active site is "controlled" by the Glu113-Arg227 distance (Figure 49 B). MD simulation trajectory analysis also indicated that the Mcs1 active site is completed blocked when the Glu113-Arg227 distance is less than 12.5 Å. During the flexibility simulation, the Mcs1 active site was completely closed (Glu113-Arg227 distance < 12.5 Å) ~50% of the time and only moderately accessible (Glu113-Arg227 distance = 12.5-15 Å) the other ~50% (Figure 42 A). In very few instances was the Mcs1 cleft determined completely open (Glu113-Arg227 distance > 20 Å) (Figure 42 A).

The conformational flexibility of the subdomains and their contributions to the overall Mcs1 flexibility was also investigated (Figure 42 B). The C-terminal subdomain was far more rigid than that of the N-terminal subdomain (Figure 42 C and D). The overall root mean square deviation (RMSD) of the entire Mcs1 catalytic domain trended well with the RMSD of the N-terminal domain, indicating the N-terminal subdomain is the driving force behind Mcs1's conformational changes (Figure 42 B and C).

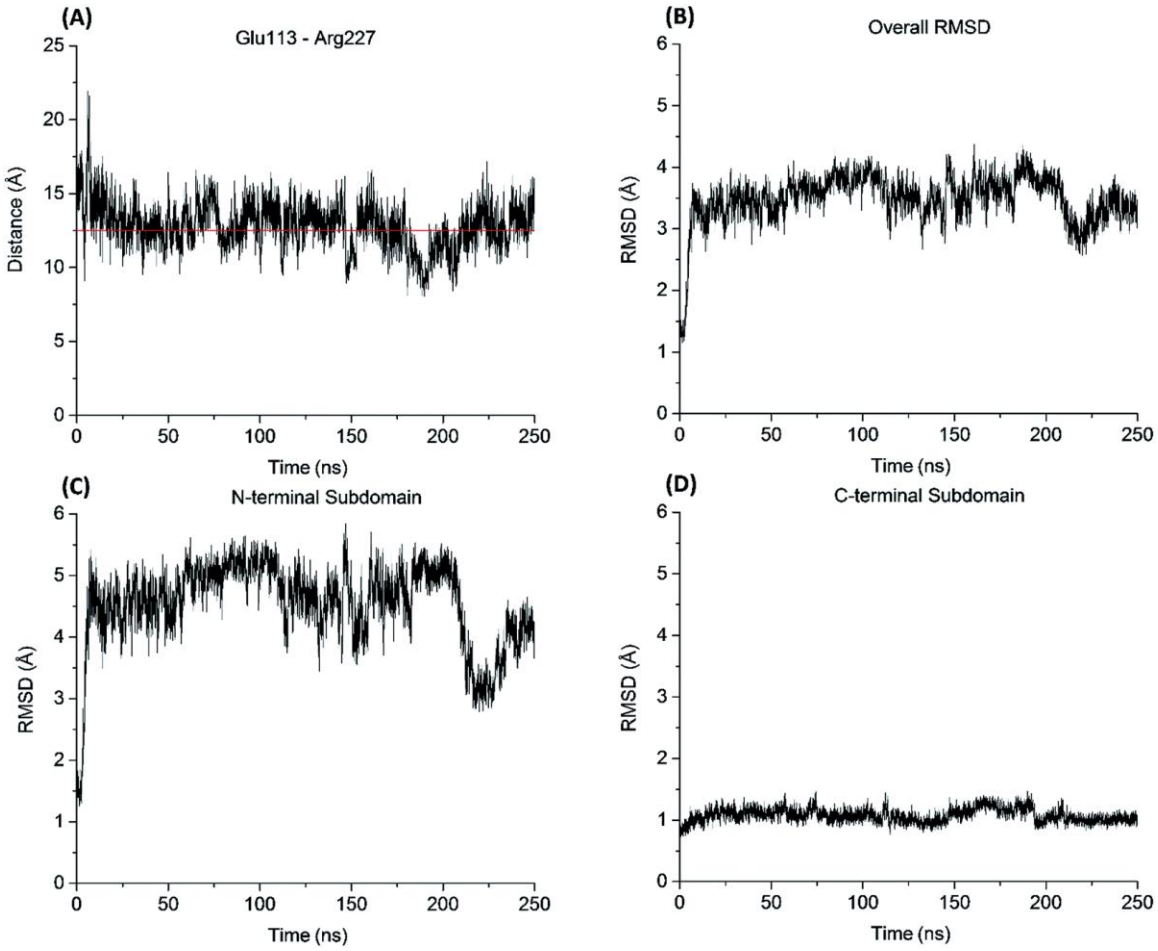


Figure 42 Mcs1 conformational dynamics computed from the 250 ns MD simulation. (A) Glu113-Arg227 residue distance; (B) overall Mcs1 structural flexibility measured in RMSD; (C) N-terminal subdomain structural flexibility measured in RMSD; (D) C-terminal subdomain structural flexibility measured in RMSD. Cα atoms were used in the residue distance and RMSD calculations.

III.4 VCAM-1 Homology Modelling

The intact VCAM-1 protein has seven immunoglobulin-like domains, but only a few fragments of the full-length VCAM-1 structure are available. D1D2 domain of VCAM-1 was well characterized and structurally determined, but previous structure studies merely indicate the D4D5 domain has a similar structure and functions to the D1D2 domain (Jones et al., 1995; Wang et al., 1995). The full-length VCAM-1 crystal structure is not available regardless of a lot of research studies of VCAM-1 have been done. Under this circumstance, a full-length VCAM-1 model is essential and urgent for related research.

The experimental result shows the cleavage site is at 42 kDa out of an entire 94 kDa of VCAM-1 amino acid sequence (M. J. Aldape et al., 2017). Accordingly, in the entire amino acid sequence of VCAM-1, the cleavage site is located at the D4D5 domain. From *Clostridium histolyticum* which is the same genus but different species from *C. sordellii*, Asp229 of clostripain is responsible for binding Arg at the P1 position (Labrou & Rigden, 2004). As shown in Figure 50, Arg381 at the potential cleavage site of the D4D5 domain is highly responsible for the binding of Mcs1. The amino acid from the N-terminus to the Arg381 has a weight of 41946.8 Da, highly consistent with the experimental data.

D4D5 domain shares 66% identity and 100% query coverage with the D1D2 domain of VCAM-1. Thus, using the D1D2 domain as a template, we built a homology model for the D4D5 domain. We performed a Protein-Blast search on the NCBI server to get the model template for the query sequence. Prime module (Jacobson, Friesner, Xiang, & Honig, 2002; Jacobson et al., 2004) of Schrodinger Suite was used to build the

homology model for the query sequence based on the template structure. We then performed the homology modeling for the full-length VCAM-1 with the following procedure in VMD (Humphrey, Dalke, & Schulten, 1996) for alignment and visualization: 1) fix VCAM-1 D1D2 domain in space; 2) align the D1 domain of a second VCAM-1 D1D2 molecule to the D2 domain of the initial VCAM-1 D1D2 molecule; 3) repeat step 2 five times to elongate the molecule to fulfill seven domains; 4) align VCAM-1 D3, D4D5 and D6D7 domains to the D2 domain of the second VCAM-1 D1D2 molecule, D1D2 of the forth VCAM-1 D1D2 molecule and D1D2 of the sixth VCAM-1 D1D2 molecule, respectively; 4) in Schrodinger Maestro, connect and fuse D1D2, D3, D4D5 and D6D7 domains of VCAM-1 to get the full-length of VCAM-1; 5) use Prime to minimize VCAM-1 molecule.

The full-length VCAM-1 structure as shown in Figure 43 was fused with the crystal structure of the D1D2 and homology model of D3, D4D5, D6, and D7. Each domain of VCAM-1 has at least one intra-domain disulfide bond to maintain the integrity of the structure. D1 or D4 domain has an extra intra-domain disulfide bond, thus this additional feature makes them distinct from other domains. In the full-length model, located at the disordered linker of the D4D5 domain, exposure of Arg381 makes it easy for binding and cleavage. There are seven Arginine residues located at the D1D2 domain, but none of them is located at the linker. Arg381 makes D4D5 distinct from D1D2 even though they are they share 66% identity and 100% query coverage.

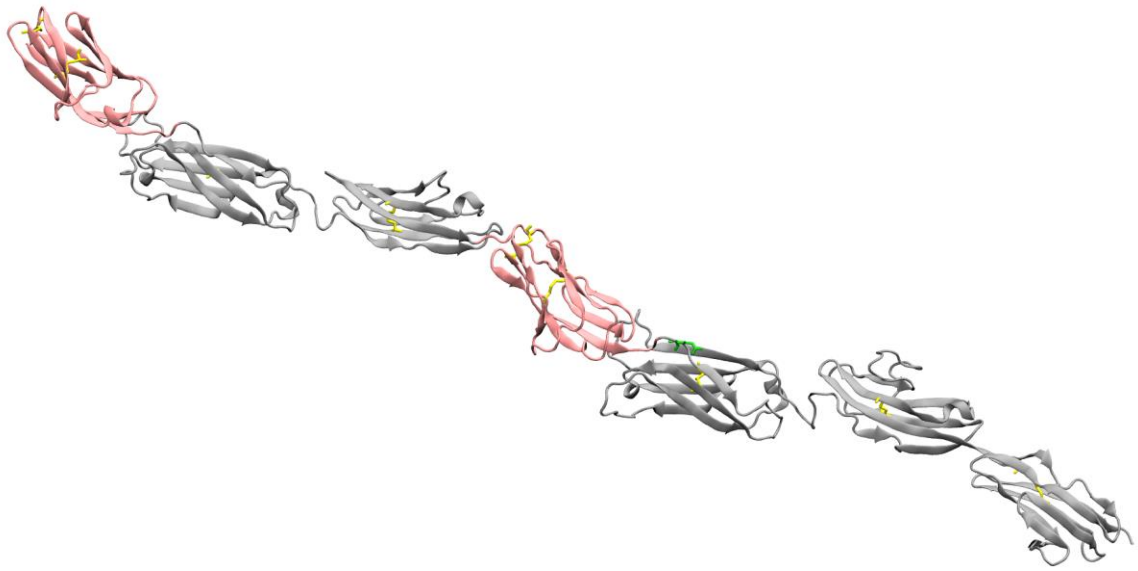


Figure 43 VCAM-1 model. D1 and D4 domains (pink) have two disulfide bonds (yellow), respectively. D2, D3, D5, D6, and D7 domains (white) have one disulfide bond (yellow), respectively. Arg381 (green) is located at the D4D5 domain.

III.5 VCAM-1 Molecular Dynamics Simulation

100 ns MD simulations were performed on D4D5 and full-length of the VCAM-1 model, respectively, with the same MD protocol as Mcs1. During the 100 ns MD simulation, the full-length VCAM-1 bent and had undergone a large-scale conformational change. Since Arg381 is the key potential residue for Mcs1 binding, the exposure of Arg381 could be the decisive factor in Mcs1 binding. We used Arg381 as the vertex and the residues at the two ends of the D4D5 domain as the points on each side of the angle. The change of the angle showed the D4D5 domain was more bent in the full-length VCAM-1 than by itself (Figure 44 A&B). In addition, the distance between Gly304 in the D4 domain and Arg381 can also determine the exposure of Arg381. As shown in Figure 44 C&D, the change of distance suggests Arg381 was less interfered with by Gly304 in full-length VCAM-1 than in the D4D5 domain. The change of angle and distance of VCAM-1 suggests the D4D5 domain in full-length VCAM-1 is more active and closer to the real environment of cleavage activity. Thus, the most bent D4D5, which is the most exposed Arg381 in the D4D5 domain of full-length VCAM-1 in the trajectory of the MD simulation, was extracted for protein-protein docking.

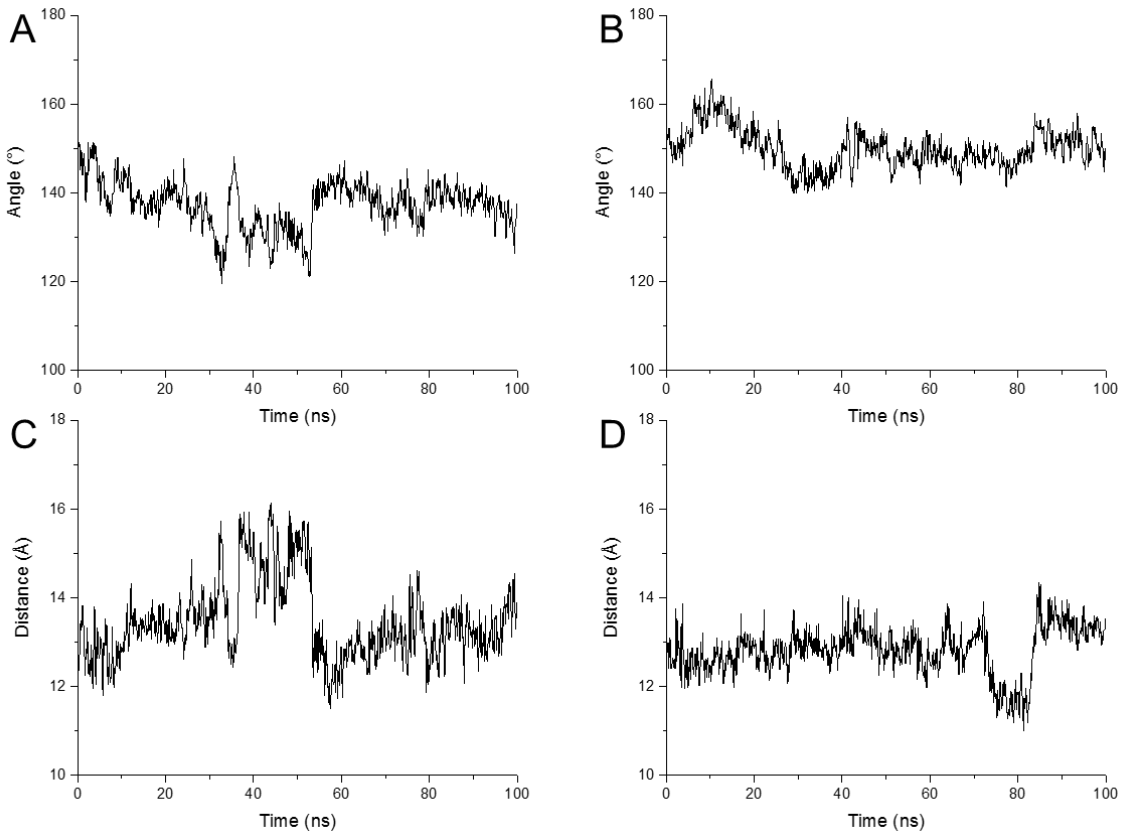


Figure 44 Distance and angle of D4D5 domain during 100 ns MD simulation.

(A) Angle of Val291-Arg381-Asn484 D4D5 domain of full-length VCAM-1 during 100 ns MD simulation.

(B) Angle of Val291-Arg381-Asn484 in the D4D5 domain during 100 ns MD simulation.

(C) Distance between Arg381 and Gly304 in the D4D5 domain of full-length VCAM-1 during 100 ns MD simulation.

(D) Distance between Arg381 and Gly304 in the D4D5 domain during 100 ns MD simulation.

III.6 Protein-Protein Docking Study

Protein-protein docking was performed by Megadock (Ohue et al., 2014), with the most open Mcs1 and the most bent D4D5 of the full-length VCAM-1. We set voxel at 0.6, Mcs1 as the receptor, and VCAM-1 D4D5 domain as the ligand in Megadock 4.0 for protein-protein docking. Among the 2,000 docking poses, the top-scored pose indicates the binding mode of Mcs1 and VCAM-1 as shown in Figure 45. Sitting in the bottom of the Mcs1 cleft, the D4D5 domain perfectly docked to Mcs1. Figure 51 shows that Arg381 of VCAM-1 is encompassed by the Zinc-Glu113-Arg227 triangle of Mcs1.

As shown in Figure 52, the electrostatic potential surface of the binding site of Mcs1 is negatively charged while the binding site of VCAM-1 is positively charged. The docking result shows that both the shape and electrostatic state of the binding surface of the two binding partners contribute to the binding and catalysis of VCAM-1.

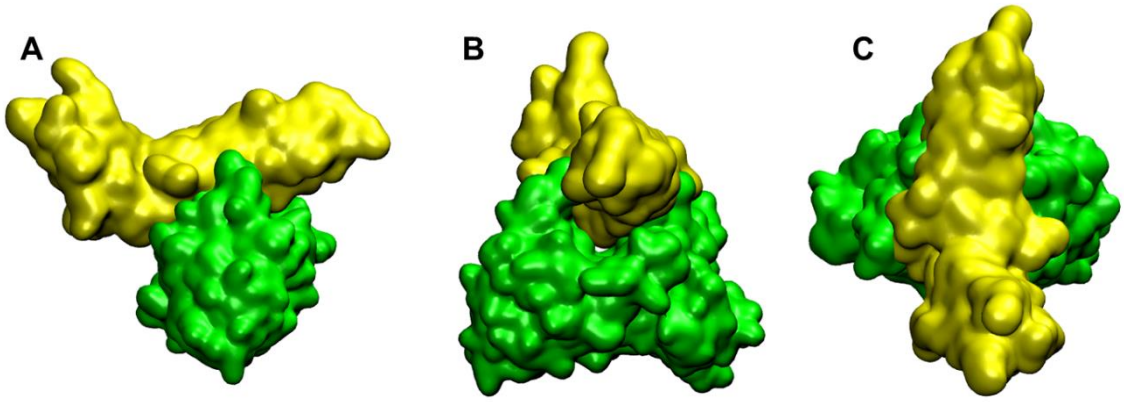


Figure 45 Best docking pose of Mcs1 (green) and VCAM-1 (yellow) from different perspectives, side view (A), front view (B), and high-angle shot (C).

III.7 High Throughput Virtual Screening Study

Using docking software packages Glide (Glide, 2018-4b), AutoDock Vina (Trott & Olson, 2010), and MVD (Molegro, 2011), high-throughput virtual screenings were applied to protein-ligand docking. 2,016 compounds from the National Cancer Institute Diversity Set III were docked to the binding surface of Mcs1. Three different conformations of Mcs1 were used in the docking study, which was the initial built, the most open, and the most closed Mcs1 structures.

Docking scores from MVD, Vina docking score, and Glide and Emodel scores from Schrodinger Glide of each ligand were collected and analyzed. Because of the different algorithms were used in those four different scoring functions, the scores cannot be compared directly between different scoring methods. However, the ranking of compounds was inter-comparable with different scoring methods. Thus a consensus scoring method was applied in the final scoring function. The rankings of every compound from those four different scoring methods were added arithmetically without weighting. The summations of all the compound rankings were re-ranked and assigned in a consecutive and ascending order. The top-ranked 40 available compounds were thus ordered for the further investigation of the Mcs1 inhibition study (the top 40 docking result data shown in **Error! Reference source not found.**).

The 40 compounds were then evaluated by fluorescence resonance energy transfer (FRET) study with Mcs1 protease assay in Dr. Michael Aldape's laboratory. All the 40 FRET results were shown in **Error! Reference source not found.**. Among the top 40 c ompounds, #227186, #639174, #61610, #67436, #177365, #71881, and #91529 were

confirmed with good inhibition of Mcs1 proteolytic activity in FRET assay (shown in Figure 47).

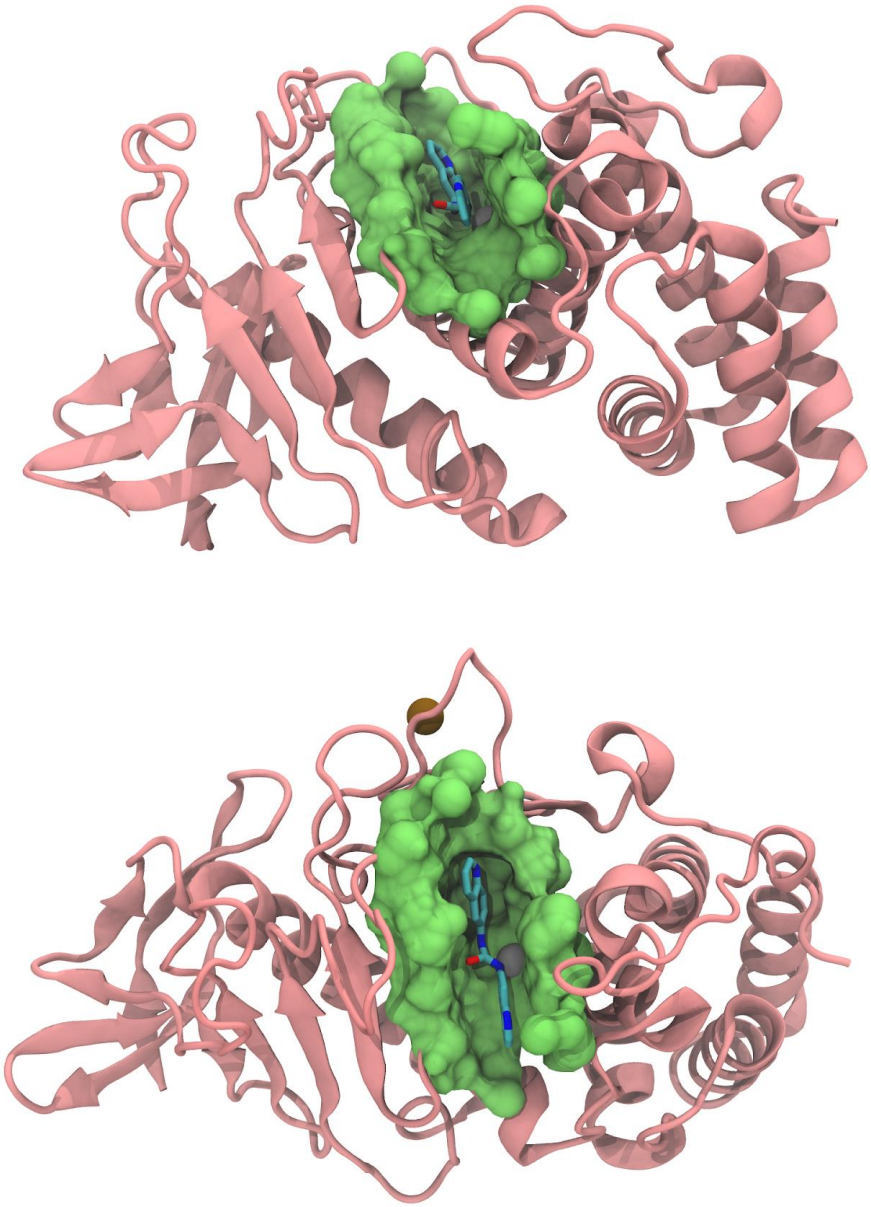


Figure 46 Docking pose of compound #71881 (in licorice representation) with Mcs1 protease (pink ribbon represents protein and green surface represents the binding site, white sphere represents Zinc ion, and brown sphere represents Calcium ion) in front view (top) and top view (bottom).

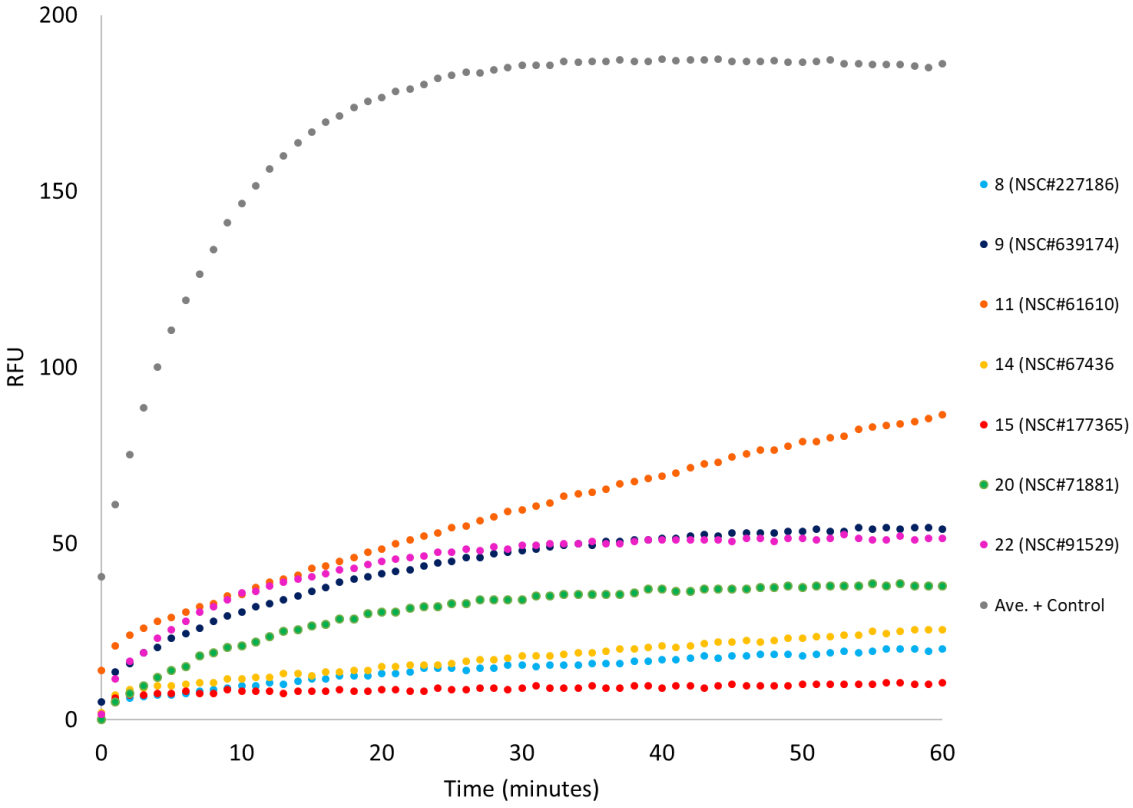


Figure 47 Best seven FRET results of the top 40 Mcs1 inhibitors.

III.8 Conclusion

The experimental data of Aldape et al. showed a fragment of VCAM-1 with around 42k Da molecular weight was cleaved by Mcs1 in vitro (M. J. Aldape et al., 2017). Only one single band was observed in the VCAM-1 cleavage assay with the presence of Mcs1, which indicates only one cleavage site exists in VCAM-1 and only one form of VCAM-1 fragment was cleaved in the experiment. Although VCAM-1 could have two different forms which are seven domains in full-length or six domains with alternatively spliced (Myron I Cybulsky et al., 2001; Vonderheide, Tedder, Springer, & Staunton, 1994), the experimental result suggests only seven domain VCAM-1 was predominant or Mcs1 could only cleave the seven domain VCAM-1 (M. J. Aldape et al., 2017).

Our docking result shows the Arg381 of VCAM-1 is close to the Mcs1 catalytic site. With a molecular weight of 41946.8 Da, the fragment from N-terminal amino acid to Arg381 of VCAM-1 can be cleaved by Mcs1 at Arg381. The electrostatic surface of the two binding partners also demonstrates the binding preference which Arg381 prefers to bind Mcs1 at the catalytic site. From the MD simulation study, we were able to sample more conformations of Mcs1 and VCAM-1 proteins. In this study, MD simulation shows large-scale conformational changes of full-length VCAM-1 which was reported in experiments. In the high-throughput virtual screening study, some of the compounds showed good binding affinity to Mcs1, which was also confirmed in the experimental study. We could further investigate and design the inhibitors of the protein-protein interaction of Mcs1 and VCAM-1, to inhibit the proteolytic function of Mcs1.

References

Agarwal, S., Dugar, D., & Sengupta, S. (2010). Ranking Chemical Structures for Drug Discovery: A New Machine Learning Approach. *Journal of chemical information and modeling, 50*(5), 716-731. doi:10.1021/ci9003865

Aldape, M. J., Bryant, A. E., Ma, Y., & Stevens, D. L. (2007). The Leukemoid Reaction in Clostridium sordellii Infection: Neuraminidase Induction of Promyelocytic Cell Proliferation. *The Journal of Infectious Diseases, 195*(12), 1838-1845. doi:10.1086/518004

Aldape, M. J., Tao, A., Heeney, D. D., McIndoo, E. R., French, J. M., & Xu, D. (2017). Experimental identification and computational characterization of a novel extracellular metalloproteinase produced by Clostridium sordellii. *RSC Adv, 7*(23), 13928-13938. doi:10.1039/c6ra27654g

Altae-Tran, H., Ramsundar, B., Pappu, A. S., & Pande, V. (2017). Low Data Drug Discovery with One-Shot Learning. *ACS Cent Sci, 3*(4), 283-293. doi:10.1021/acscentsci.6b00367

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research, 25*(17), 3389-3402.

Anderson, A. C. (2003). The Process of Structure-Based Drug Design. *Chemistry & Biology, 10*(9), 787-797. doi:https://doi.org/10.1016/j.chembiol.2003.09.002

Ballesteros, A., Fenollar-Ferrer, C., & Swartz, K. J. (2018). Structural relationship between the putative hair cell mechanotransduction channel TMC1 and TMEM16 proteins. *Elife, 7*. doi:10.7554/eLife.38433

Banbula, A., Potempa, J., Travis, J., Fernandez-Catalén, C., Mann, K., Huber, R., . . . Medrano, F. (1998). Amino-acid sequence and three-dimensional structure of the Staphylococcus aureus metalloproteinase at 1.72 Å resolution. *Structure, 6*(9), 1185-1193.

Banks, J. L., Beard, H. S., Cao, Y., Cho, A. E., Damm, W., Farid, R., . . . Maple, J. R. (2005). Integrated modeling program, applied chemical theory (IMPACT). *Journal of Computational Chemistry, 26*(16), 1752-1780.

Bateman, A., Coin, L., Durbin, R., Finn, R. D., Hollich, V., Griffiths-Jones, S., . . . Sonnhammer, E. L. (2004). The Pfam protein families database. *Nucleic acids research, 32*(suppl_1), D138-D141.

Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Sayers, E. W. (2012). GenBank. *Nucleic acids research, 41*(D1), D36-D42.

Bolton, E. E., Wang, Y., Thiessen, P. A., & Bryant, S. H. (2008). PubChem: integrated platform of small molecules and biological activities *Annual reports in computational chemistry* (Vol. 4, pp. 217-241): Elsevier.

Brummett, R. E., & Fox, K. E. (1989). Aminoglycoside-induced hearing loss in humans. *Antimicrobial Agents and Chemotherapy, 33*(6), 797-800. doi:10.1128/aac.33.6.797

Cepeda, V., Fuertes, M. A., Castilla, J., Alonso, C., Quevedo, C., & Pérez, J. M. (2007). Biochemical mechanisms of cisplatin cytotoxicity. *Anti-Cancer Agents in Medicinal Chemistry (Formerly Current Medicinal Chemistry-Anti-Cancer Agents), 7*(1), 3-18.

Cheng, T., Li, Q., Zhou, Z., Wang, Y., & Bryant, S. H. (2012). Structure-Based Virtual Screening for Drug Discovery: a Problem-Centric Review. *The AAPS Journal, 14*(1), 133-141. doi:10.1208/s12248-012-9322-0

Chirtes, F., & Albu, S. (2014). Prevention and Restoration of Hearing Loss Associated with the Use of Cisplatin. *BioMed Research International, 2014*, 9. doi:10.1155/2014/925485

Choi, Y.-M., Kim, H.-K., Shim, W., Anwar, M. A., Kwon, J.-W., Kwon, H.-K., . . . Choi, S. (2015). Mechanism of Cisplatin-Induced Cytotoxicity Is Correlated to Impaired Metabolism Due to Mitochondrial ROS Generation. *PLoS ONE, 10*(8), e0135083. doi:10.1371/journal.pone.0135083

Consortium, U. (2014). UniProt: a hub for protein information. *Nucleic acids research, 43*(D1), D204-D212.

Corey, D. P., Akyuz, N., & Holt, J. R. (2019). Function and Dysfunction of TMC Channels in Inner Ear Hair Cells. *Cold Spring Harb Perspect Med, 9*(10). doi:10.1101/cshperspect.a033506

Cunningham, C. L., & Muller, U. (2019). Molecular Structure of the Hair Cell Mechanoelectrical Transduction Complex. *Cold Spring Harb Perspect Med, 9*(5). doi:10.1101/cshperspect.a033167

Cybulsky, M. I., Fries, J. W., Williams, A. J., Sultan, P., Eddy, R., Byers, M., . . . Collins, T. (1991). Gene structure, chromosomal location, and basis for alternative mRNA splicing of the human VCAM1 gene. *Proceedings of the National Academy of Sciences, 88*(17), 7859-7863.

Cybulsky, M. I., Iiyama, K., Li, H., Zhu, S., Chen, M., Iiyama, M., . . . Milstone, D. S. (2001). A major role for VCAM-1, but not ICAM-1, in early atherosclerosis. *The Journal of clinical investigation, 107*(10), 1255-1262.

Davis, B. D. (1987). Mechanism of bactericidal action of aminoglycosides. *Microbiological reviews, 51*(3), 341-350.

Desmond. (2018-4). Schrödinger Release 2018-4: Desmond, Schrödinger, LLC, New York, NY, 2018.

Dixon, S. L., Duan, J., Smith, E., Von Bargen, C. D., Sherman, W., & Repasky, M. P. (2016). AutoQSAR: an automated machine learning tool for best-practice quantitative structure–activity relationship modeling. *Future medicinal chemistry, 8*(15), 1825-1839.

Essmann, U., Perera, L., Berkowitz, M. L., Darden, T., Lee, H., & Pedersen, L. G. (1995). A smooth particle mesh Ewald method. *The Journal of chemical physics, 103*(19), 8577-8593.

Farmer, L. J., Ledeboer, M. W., Hoock, T., Arnost, M. J., Bethiel, R. S., Bennani, Y. L., . . . Zuccola, H. J. (2015). Discovery of VX-509 (Decernotinib): A Potent and Selective Janus Kinase 3 Inhibitor for the Treatment of Autoimmune Diseases. *Journal of medicinal chemistry, 58*(18), 7195-7216. doi:10.1021/acs.jmedchem.5b00301

Forli, S., Huey, R., Pique, M. E., Sanner, M. F., Goodsell, D. S., & Olson, A. J. (2016). Computational protein–ligand docking and virtual drug screening with the AutoDock suite. *Nature Protocols, 11*(5), 905.

Fragoulis, G. E., McInnes, I. B., & Siebert, S. (2019). JAK-inhibitors. New players in the field of immune-mediated diseases, beyond rheumatoid arthritis. *Rheumatology, 58*(Supplement_1), i43-i54. doi:10.1093/rheumatology/key276

Fuertes, M., Castilla, J., Alonso, C., & Prez, J. (2003). Cisplatin biochemical mechanism of action: from cytotoxicity to induction of cell death through interconnections between apoptotic and necrotic pathways. *Current medicinal chemistry, 10*(3), 257-266.

Gaulton, A., Bellis, L. J., Bento, A. P., Chambers, J., Davies, M., Hersey, A., . . . Al-Lazikani, B. (2012). ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic acids research, 40*(D1), D1100-D1107.

Glide. (2018-4a). Schrödinger Release 2018-4: AutoQSAR, Schrödinger, LLC, New York, NY, 2018.

Glide. (2018-4b). Schrödinger Release 2018-4: Glide, Schrödinger, LLC, New York, NY, 2018.

Glide. (2018-4c). Schrödinger Release 2018-4: SiteMap, Schrödinger, LLC, New York, NY, 2018.

Global Burden of Disease Study, C. (2015). Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990-2013: a systematic analysis for the Global Burden of Disease Study 2013. *Lancet (London, England), 386*(9995), 743-800. doi:10.1016/S0140-6736(15)60692-4

Gutierrez, J. A., Luo, M., Singh, V., Li, L., Brown, R. L., Norris, G. E., . . . Painter, G. F. (2007). Picomolar inhibitors as transition-state probes of 5′-methylthioadenosine nucleosidases. *ACS chemical biology, 2*(11), 725-734.

Halgren, T. A. (2009). Identifying and Characterizing Binding Sites and Assessing Druggability. *Journal of chemical information and modeling, 49*(2), 377-389. doi:10.1021/ci800324m

Hazlitt, R. A., Teitz, T., Bonga, J. D., Fang, J., Diao, S., Iconaru, L., . . . Zuo, J. (2018). Development of Second-Generation CDK2 Inhibitors for the Prevention of Cisplatin-Induced Hearing Loss. *Journal of medicinal chemistry, 61*(17), 7700-7709. doi:10.1021/acs.jmedchem.8b00669

Heikamp, K., & Bajorath, J. (2014). Support vector machines for drug discovery. *Expert Opinion on Drug Discovery, 9*(1), 93-104. doi:10.1517/17460441.2014.866943

Hessler, G., & Baringhaus, K.-H. (2018). Artificial intelligence in drug design. *Molecules, 23*(10), 2520.

Holt, J. R., & Géléoc, G. S. G. (2017). Auditory Hair Cells and Sensory Transduction: Oxford University Press.

Huang, N., Shoichet, B. K., & Irwin, J. J. (2006). Benchmarking Sets for Molecular Docking. *Journal of medicinal chemistry, 49*(23), 6789-6801. doi:10.1021/jm0608356

Humphrey, W., Dalke, A., & Schulten, K. (1996). VMD: visual molecular dynamics. *Journal of molecular graphics, 14*(1), 33-38.

Hutchin, T., & Cortopassi, G. (1994). Proposed molecular and cellular mechanism for aminoglycoside ototoxicity. *Antimicrobial Agents and Chemotherapy, 38*(11), 2517-2520. doi:10.1128/aac.38.11.2517

Ihlenfeldt, W. D., Bolton, E. E., & Bryant, S. H. (2009). The PubChem chemical structure sketcher. *Journal of cheminformatics, 1*(1), 20.

Jacobson, M. P., Friesner, R. A., Xiang, Z., & Honig, B. (2002). On the Role of the Crystal Environment in Determining Protein Side-chain Conformations. *Journal of molecular biology, 320*(3), 597-608. doi:https://doi.org/10.1016/S0022-2836(02)00470-9

Jacobson, M. P., Pincus, D. L., Rapp, C. S., Day, T. J. F., Honig, B., Shaw, D. E., & Friesner, R. A. (2004). A hierarchical approach to all-atom protein loop prediction. *Proteins: Structure, Function, and Bioinformatics, 55*(2), 351-367. doi:10.1002/prot.10613

Jaeger, S., Fulle, S., & Turk, S. (2018). Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition. *Journal of chemical information and modeling, 58*(1), 27-35. doi:10.1021/acs.jcim.7b00616

Jiang, C., Jin, X., Dong, Y., & Chen, M. (2016). Kekule. js: an open source javascript chemoinformatics toolkit. *Journal of chemical information and modeling, 56*(6), 1132-1138.

Jiménez, J., Doerr, S., Martínez-Rosell, G., Rose, A. S., & De Fabritiis, G. (2017). DeepSite: protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics, 33*(19), 3036-3042.

Johnson, M., Zaretskaya, I., Raytselis, Y., Merezhuk, Y., McGinnis, S., & Madden, T. L. (2008). NCBI BLAST: a better web interface. *Nucleic acids research, 36*(suppl_2), W5-W9.

Jones, E. Y., Harlos, K., Bottomley, M. J., Robinson, R. C., Driscoll, P. C., Edwards, R. M., . . . Stuart, D. I. (1995). Crystal structure of an integrin-binding fragment of vascular cell adhesion molecule-1 at 1.8 A resolution. *Nature, 373*(6514), 539-544.

Jorgensen, W. L., Maxwell, D. S., & Tirado-Rives, J. (1996). Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *Journal of the American Chemical Society, 118*(45), 11225-11236. doi:10.1021/ja9621760

Kapetanovic, I. M. (2008). Computer-aided drug discovery and development (CADDD): in silico-chemico-biological approach. *Chemico-biological interactions, 171*(2), 165-176. doi:10.1016/j.cbi.2006.12.006

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., . . . Bolton, E. E. (2019). PubChem 2019 update: improved access to chemical data. *Nucleic acids research, 47*(D1), D1102-D1109. doi:10.1093/nar/gky1033

Kimura, A. C., Higa, J. I., Levin, R. M., Simpson, G., Vargas, Y., & Vugia, D. J. (2004). Outbreak of Necrotizing Fasciitis Due to Clostridium sordellii among Black-Tar Heroin Users. *Clinical Infectious Diseases, 38*(9), e87-e91. doi:10.1086/383471

Koes, D. R., Baumgartner, M. P., & Camacho, C. J. (2013). Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. *Journal of chemical information and modeling, 53*(8), 1893-1904.

Koes, D. R., & Camacho, C. J. (2012). ZINCPharmer: pharmacophore search of the ZINC database. *Nucleic acids research, 40*(W1), W409-W414.

Konc, J., & Janežič, D. (2014). Binding site comparison for function prediction and pharmaceutical discovery. *Current opinion in structural biology, 25*, 34-39. doi:https://doi.org/10.1016/j.sbi.2013.11.012

Kuhl, H. (2005). Pharmacology of estrogens and progestogens: influence of different routes of administration. *Climacteric, 8*(sup1), 3-63. doi:10.1080/13697130500148875

Kurima, K., Ebrahim, S., Pan, B., Sedlacek, M., Sengupta, P., Millis, B. A., . . . Kachar, B. (2015). TMC1 and TMC2 Localize at the Site of Mechanotransduction in Mammalian Inner Ear Hair Cell Stereocilia. *Cell reports, 12*(10), 1606-1617. doi:10.1016/j.celrep.2015.07.058

Labrou, N. E., & Rigden, D. J. (2004). The structure-function relationship in the clostripain family of peptidases. *Eur J Biochem, 271*(5), 983-992.

Landrum, G. (2006). RDKit: Open-source cheminformatics.

Lanvers-Kaminsky, C., Zehnhoff-Dinnesen, A. a., Parfitt, R., & Ciarimboli, G. (2017). Drug-induced ototoxicity: Mechanisms, Pharmacogenetics, and protective strategies. *Clinical Pharmacology & Therapeutics, 101*(4), 491-500. doi:10.1002/cpt.603

Lavecchia, A. (2015). Machine-learning approaches in drug discovery: methods and applications. *Drug discovery today, 20*(3), 318-331. doi:https://doi.org/10.1016/j.drudis.2014.10.012

Le Guilloux, V., Schmidtke, P., & Tuffery, P. (2009). Fpocket: an open source platform for ligand pocket detection. *BMC bioinformatics, 10*(1), 168.

Lee, J. E., Singh, V., Evans, G. B., Tyler, P. C., Furneaux, R. H., Cornell, K. A., . . . Howell, P. L. (2005). Structural rationale for the affinity of pico-and femtomolar transition state analogues of Escherichia coli 5′-methylthioadenosine/S-adenosylhomocysteine nucleosidase. *Journal of Biological Chemistry, 280*(18), 18274-18282.

Li, H., Robertson, A. D., & Jensen, J. H. (2005). Very fast empirical prediction and rationalization of protein pKa values. *Proteins: Structure, Function, and Bioinformatics, 61*(4), 704-721.

Lo, Y.-C., Rensi, S. E., Torng, W., & Altman, R. B. (2018). Machine learning in chemoinformatics and drug discovery. *Drug discovery today, 23*(8), 1538-1546. doi:https://doi.org/10.1016/j.drudis.2018.05.010

Lomize, M. A., Pogozheva, I. D., Joo, H., Mosberg, H. I., & Lomize, A. L. (2012). OPM database and PPM web server: resources for positioning of proteins in membranes. *Nucleic acids research, 40*(D1), D370-D376.

Macalino, S. J. Y., Gosu, V., Hong, S., & Choi, S. (2015). Role of computer-aided drug design in modern drug discovery. *Archives of Pharmacal Research, 38*(9), 1686-1701. doi:10.1007/s12272-015-0640-5

Maestro. (2018-4). Schrödinger Release 2018-4: Maestro, Schrödinger, LLC, New York, NY, 2018.

Manallack, D. T., & Livingstone, D. J. (1999). Neural networks in drug discovery: have they lived up to their promise? *European Journal of Medicinal Chemistry*, *34*(3), 195-208. doi:https://doi.org/10.1016/S0223-5234(99)80052-X

Marrin, C. (2011). Webgl specification. *Khronos WebGL Working Group*, *3*.

Marshall, G. R. (1987). Computer-aided drug design. *Annual review of pharmacology and toxicology*, *27*(1), 193-213.

Martínez-Rosell, G., Giorgino, T., & De Fabritiis, G. (2017). PlayMolecule ProteinPrepare: a web application for protein preparation for molecular dynamics simulations. *Journal of chemical information and modeling*, *57*(7), 1511-1516.

Martyna, G. J., Klein, M. L., & Tuckerman, M. (1992). Nosé–Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of chemical physics*, *97*(4), 2635-2643.

Matsumoto, A., Aoki, S., & Ohwada, H. (2016). Comparison of random forest and SVM for raw data in drug discovery: prediction of radiation protection and toxicity case study. *International Journal of Machine Learning and Computing*, *6*(2), 145.

Merz Jr, K. M., Ringe, D., & Reynolds, C. H. (2010). *Drug design: structure-and ligand-based approaches*: Cambridge University Press.

Milletti, F., & Vulpetti, A. (2010). Predicting Polypharmacology by Binding Site Similarity: From Kinases to the Protein Universe. *Journal of chemical information and modeling*, *50*(8), 1418-1431. doi:10.1021/ci1001263

Mingeot-Leclercq, M.-P., Glupczynski, Y., & Tulkens, P. M. (1999). Aminoglycosides: Activity and Resistance. *Antimicrobial Agents and Chemotherapy*, *43*(4), 727-737. doi:10.1128/aac.43.4.727

Molegro, A. (2011). MVD 5.0 Molegro Virtual Docker. *DK-8000 Aarhus C, Denmark*.

Monzani, D., Galeazzi, G. M., Genovese, E., Marrara, A., & Martini, A. (2008). Psychological profile and social behaviour of working adults with mild or moderate hearing loss. *Acta otorhinolaryngologica Italica : organo ufficiale della Societa italiana di otorinolaringologia e chirurgia cervico-facciale, 28*(2), 61-66.

Neudert, G., & Klebe, G. (2011). fconv: Format conversion, manipulation and feature computation of molecular data. *Bioinformatics, 27*(7), 1021-1022.

Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology, 24*(12), 1565-1567.

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., & Hutchison, G. R. (2011). Open Babel: An open chemical toolbox. *Journal of cheminformatics, 3*(1), 33.

O'Sullivan, M. E., Perez, A., Lin, R., Sajjadi, A., Ricci, A. J., & Cheng, A. G. (2017). Towards the Prevention of Aminoglycoside-Related Hearing Loss. *Frontiers in Cellular Neuroscience, 11*(325). doi:10.3389/fncel.2017.00325

Ohue, M., Shimoda, T., Suzuki, S., Matsuzaki, Y., Ishida, T., & Akiyama, Y. (2014). MEGADOCK 4.0: an ultra–high-performance protein–protein docking software for heterogeneous supercomputers. *Bioinformatics, 30*(22), 3281-3283.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). *How many trees in a random forest?* Paper presented at the International workshop on machine learning and data mining in pattern recognition.

Pan, B., Akyuz, N., Liu, X. P., Asai, Y., Nist-Lund, C., Kurima, K., . . . Holt, J. R. (2018). TMC1 Forms the Pore of Mechanosensory Transduction Channels in

Vertebrate Inner Ear Hair Cells. *Neuron, 99*(4), 736-753 e736. doi:10.1016/j.neuron.2018.07.033

Pirhadi, S., Sunseri, J., & Koes, D. R. (2016). Open source molecular modeling. *Journal of Molecular Graphics and Modelling, 69*, 127-143.

Pussegoda, K., Ross, C. J., Visscher, H., Yazdanpanah, M., Brooks, B., Rassekh, S. R., . . . Consortium, C. (2013). Replication of TPMT and ABCC3 Genetic Variants Highly Associated With Cisplatin-Induced Hearing Loss in Children. *Clinical Pharmacology & Therapeutics, 94*(2), 243-251. doi:10.1038/clpt.2013.80

Ramirez, N., & Abel-Santos, E. (2010). Requirements for Germination of Clostridium sordellii Spores <em>In Vitro</em&gt. *Journal of Bacteriology, 192*(2), 418. doi:10.1128/JB.01226-09

Reddy, A. S., & Zhang, S. (2013). Polypharmacology: drug discovery for the future. *Expert Review of Clinical Pharmacology, 6*(1), 41-47. doi:10.1586/ecp.12.74

Rogers, D., & Hahn, M. (2010). Extended-Connectivity Fingerprints. *Journal of chemical information and modeling, 50*(5), 742-754. doi:10.1021/ci100050t

Rose, A. S., & Hildebrand, P. W. (2015). NGL Viewer: a web application for molecular visualization. *Nucleic acids research, 43*(W1), W576-W579.

Rose, P. W., Beran, B., Bi, C., Bluhm, W. F., Dimitropoulos, D., Goodsell, D. S., . . . Westbrook, J. D. (2010). The RCSB Protein Data Bank: redesigned web site and web services. *Nucleic acids research, 39*(suppl_1), D392-D401.

Rush, T. S., 3rd, Grant, J. A., Mosyak, L., & Nicholls, A. (2005). A shape-based 3-D scaffold hopping method and its application to a bacterial protein-protein interaction. *J Med Chem, 48*(5), 1489-1495. doi:10.1021/jm040163o

Rybak, L. P., & Ramkumar, V. (2007). Ototoxicity. *Kidney International, 72*(8), 931-935. doi:https://doi.org/10.1038/sj.ki.5002434

Schacht, J., Talaska, A. E., & Rybak, L. P. (2012). Cisplatin and Aminoglycoside Antibiotics: Hearing Loss and Its Prevention. *The Anatomical Record, 295*(11), 1837-1850. doi:10.1002/ar.22578

Schmidtke, P., Le Guilloux, V., Maupetit, J., & Tuffery, P. (2010). Fpocket: online tools for protein ensemble pocket detection and tracking. *Nucleic acids research, 38*(suppl_2), W582-W589.

Schrödinger, L. (2015). Schrödinger Release 2015-4: Desmond molecular dynamics system, Maestro-Desmond interoperability tools, DE Shaw Research. *Schrödinger, New York, NY*.

Sharma, D., Cukras, A. R., Rogers, E. J., Southworth, D. R., & Green, R. (2007). Mutational analysis of S12 protein and implications for the accuracy of decoding by the ribosome. *Journal of molecular biology, 374*(4), 1065-1076. doi:10.1016/j.jmb.2007.10.003

Shen, M. y., & Sali, A. (2006). Statistical potential for assessment and prediction of protein structures. *Protein science, 15*(11), 2507-2524.

Sheth, S., Mukherjea, D., Rybak, L. P., & Ramkumar, V. (2017). Mechanisms of Cisplatin-Induced Ototoxicity and Otoprotection. *Frontiers in Cellular Neuroscience, 11*(338). doi:10.3389/fncel.2017.00338

Sitruk-Ware, R. (2005). New progestagens for contraceptive use. *Human Reproduction Update, 12*(2), 169-178. doi:10.1093/humupd/dmi046

Skalic, M., Martínez-Rosell, G., Jiménez, J., & De Fabritiis, G. (2019). PlayMolecule BindScope: large scale CNN-based virtual screening on the web. *Bioinformatics, 35*(7), 1237-1238.

Skalic, M., Varela-Rial, A., Jiménez, J., Martínez-Rosell, G., & De Fabritiis, G. (2019). LigVoxel: inpainting binding pockets using 3D-convolutional neural networks. *Bioinformatics, 35*(2), 243-250.

Sommer, K., Friedrich, N.-O., Bietz, S., Hilbig, M., Inhester, T., & Rarey, M. (2016). UNICON: a powerful and easy-to-use compound library converter: ACS Publications.

Subramanian, G., Ramsundar, B., Pande, V., & Denny, R. A. (2016). Computational Modeling of beta-Secretase 1 (BACE-1) Inhibitors Using Ligand Based Approaches. *J Chem Inf Model, 56*(10), 1936-1949. doi:10.1021/acs.jcim.6b00290

Sunseri, J., & Koes, D. R. (2016). Pharmit: interactive exploration of chemical space. *Nucleic acids research, 44*(W1), W442-W448.

Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. (2003). Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling. *Journal of Chemical Information and Computer Sciences, 43*(6), 1947-1958. doi:10.1021/ci034160g

Svetnik, V., Liaw, A., Tong, C., & Wang, T. (2004). *Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules*, Berlin, Heidelberg.

Tao, A., Huang, Y., Shinohara, Y., Caylor, M. L., Pashikanti, S., & Xu, D. (2019). ezCADD: A Rapid 2D/3D Visualization-Enabled Web Modeling Environment for Democratizing Computer-Aided Drug Design. *Journal of chemical information and modeling*, *59*(1), 18-24. doi:10.1021/acs.jcim.8b00633

Teitz, T., Fang, J., Goktug, A. N., Bonga, J. D., Diao, S., Hazlitt, R. A., . . . Zuo, J. (2018). CDK2 inhibitors as candidate therapeutics for cisplatin- and noise-induced hearing loss. *Journal of Experimental Medicine*, *215*(4), 1187-1203. doi:10.1084/jem.20172246

Terfloth, L., & Gasteiger, J. (2001). Neural networks and genetic algorithms in drug design. *Drug discovery today, 6,* 102-108. doi:https://doi.org/10.1016/S1359-6446(01)00173-8

Thomas, A. J., Hailey, D. W., Stawicki, T. M., Wu, P., Coffin, A. B., Rubel, E. W., . . . Ou, H. C. (2013). Functional Mechanotransduction Is Required for Cisplatin-Induced Hair Cell Death in the Zebrafish Lateral Line. *The Journal of Neuroscience*, *33*(10), 4405-4414. doi:10.1523/jneurosci.3940-12.2013

Ton, C., & Parng, C. (2005). The use of zebrafish for assessing ototoxic and otoprotective agents. *Hearing Research*, *208*(1), 79-88. doi:https://doi.org/10.1016/j.heares.2005.05.005

Trott, O., & Olson, A. J. (2010). AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, *31*(2), 455-461.

Vamathevan, J., Clark, D., Czodrowski, P., Dunham, I., Ferran, E., Lee, G., . . . Zhao, S. (2019). Applications of machine learning in drug discovery and development. *Nature Reviews Drug Discovery, 18*(6), 463-477. doi:10.1038/s41573-019-0024-5

Veselovsky, A., & Ivanov, A. (2003). Strategy of computer-aided drug design. *Current Drug Targets-Infectious Disorders, 3*(1), 33-40.

Vonderheide, R. H., Tedder, T. F., Springer, T. A., & Staunton, D. E. (1994). Residues within a conserved amino acid motif of domains 1 and 4 of VCAM-1 are required for binding to VLA-4. *Journal of Cell Biology, 125*(1), 215-222. doi:10.1083/jcb.125.1.215

Wang, J.-H., Pepinsky, R. B., Stehle, T., Liu, J.-h., Karpusas, M., Browning, B., & Osborn, L. (1995). The crystal structure of an N-terminal two-domain fragment of vascular cell adhesion molecule 1 (VCAM-1): a cyclic peptide based on the domain 1 CD loop can inhibit VCAM-1-alpha 4 integrin interaction. *Proceedings of the National Academy of Sciences, 92*(12), 5714-5718.

Webb, B., & Sali, A. (2014). Comparative Protein Structure Modeling Using MODELLER. *Current Protocols in Bioinformatics, 47*(1), 5.6.1-5.6.32. doi:10.1002/0471250953.bi0506s47

Wishart, D. S., Knox, C., Guo, A. C., Shrivastava, S., Hassanali, M., Stothard, P., . . . Woolsey, J. (2006). DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic acids research, 34*(suppl_1), D668-D672.

Xu, D., & Zhang, Y. (2013). Toward optimal fragment generations for ab initio protein structure assembly. *Proteins: Structure, Function, and Bioinformatics, 81*(2), 229-239. doi:10.1002/prot.24179

Yang, J., & Zhang, Y. (2015). I-TASSER server: new development for protein structure and function predictions. *Nucleic acids research, 43*(W1), W174-W181. doi:10.1093/nar/gkv342

Appendix A

Table 10 Compound dataset (sample) for machine learning study.

Compound ID	**Activity**	**SMILES**
JZC-1	1	Oc1ccc(cc1)c2[nH]c3nnc(NC(=O)C4CC4)c3cc2Br
JZC-2	1	NS(=O)(=O)c1ccc(cc1)N=Nc2c(O)[nH]c3ccc(Br)cc23
JZC-3	1	Oc1[nH]c2cc(Br)ccc2c1c3[nH]c4ccccc4c3N=O
JZC-4	1	Brc1ccc2[nH]c3c(CC(=O)Nc4ccccc34)c2c1
JZC-5	1	COc1cccc(c1)C(=O)c2sc(Nc3ccc(cc3)N4CCN(CC4)C(C)C)nc2N
JZC-6	1	Clc1cccc(Cl)c1C(=O)Nc2c[nH]nc2C(=O)NC3CCNCC3
JZC-7	1	COc1cccc(c1)n2ncc3c(N\\N=C\c4ccncc4)ncnc23
JZC-8	1	COc1cccc(c1)c2c[nH]c3c(N\\N=C\c4ccncc4)ncnc23
JZC-9	1	CC[C@H](CO)Nc1nc(NCc2ccccc2O)c3ncn(C(C)C)c3n1
JZC-10	1	Nc1nc(NCCNc2ncc(c(n2)c3ccc(Cl)cc3Cl)n4ccnc4)ccc1[N+](=O)[O-]
JZC-11	1	CC(C)n1c(C)ncc1c2ccnc(Nc3ccc(cc3)S(=O)(=O)C)n2
JZC-12	1	[O-][N+](=O)c1ccc2[nH]c3c(CC(=O)Nc4ccccc34)c2c1
JZC-13	1	CCCCOc1c(c[nH]c2nncc12)C(=O)c3c(F)cc(C)cc3F
JZC-14	1	CCN1CCN(Cc2ccc(Nc3ncc(F)c(n3)c4cc(F)c5nc(C)n(C(C)C)c5c4)nc2)CC1
JZC-15	1	Cn1cc(C2=C(C(=O)NC2=O)c3ccc(Cl)cc3Cl)c4ccccc14
JZC-16	1	COc1cc(C=NNc2ncnc3c2cnn3c4ccccc4)ccc1O
JZC-17	1	COc1cc[nH]c1C=C2C(=O)Nc3ccc(F)c(C#CC4(O)CCNCC4)c23
JZC-18	1	COc1ccc2NC(=O)C(=Cc3c[nH]cn3)c2c1
JZC-19	1	CCCCc1c([nH]c2nccnc12)c3ccc(O)cc3
JZC-20	1	COc1ccc(OC)c(c1)C2=C(C(=O)NC2=O)c3cn(C)c4ccccc34
…	…	…
JZC-200	0	FC1=CC=CC(F)=C1C(NC2=CNN=C2C(NC3CCNCC3)=O)=O
JZC-201	0	FC1=CC(F)=CC(F)=C1C(NC2=CNN=C2C(NC3CCNCC3)=O)=O
JZC-202	0	FC1=CC=CC(OC)=C1C(NC2=CNN=C2C(NC3CCNCC3)=O)=O
JZC-203	1	FC1=CC=CC(Cl)=C1C(NC2=CNN=C2C(NC3CCNCC3)=O)=O
JZC-204	1	O=C(C1=C(Cl)C=CC=C1Cl)NC2=CNN=C2C(NC3CCN(C(C4=C(Cl)C=CC=C4Cl)=O)CC3)=O
JZC-205	0	O=C(C1=C(Cl)C=CC=C1Cl)NC2=CN(C)N=C2C(NC3CCN(C(C4=C(Cl)C=CC=C4Cl)=O)CC3)=O
JZC-206	0	O=C(NC1=CN(N=C1C(NC2=CC=CC=C2)=O)CC)C3=CC=C(C=C3Cl)Cl

Table 10 Compound dataset (sample) for machine learning study. (continue)

Compound ID	**Activity**	**SMILES**
JZC-207	1	O=C(C1=CC=C(CN2C=C(Cl)C=N2)C=C1)NC3=CN(CC)N=C3C(NC4CCCC4)=O
JZC-208	1	O=C(NC1=CN(N=C1C(NCC(C)C)=O)CC)C2=CC=C(COCC(F)(F)F)C=C2
JZC-209	0	O=C(NC1=CN(N=C1C(NCC(C)C)=O)CC)C2=C(F)C=CC=C2F
JZC-210	0	O=C(NC1=CN(N=C1C(NCC(C)C)=O)CC)C2=CC=CC=C2Cl
JZC-211	0	O=C(NC1=CN(N=C1C(NC)=O)C)C2=CC=CC=C2Cl
JZC-212	0	O=C(NC1=CN(N=C1C(NCCC2=CC=CO2)=O)C)C3=CC=C(Cl)C=C3Cl
JZC-213	0	O=C(N1CCCCC1)C2=C(NC(C3=C(Cl)C=CC=C3)=O)C=NN2C
JZC-214	0	CCCNC(C1=C(C=NN1C)NC(C2=C(Cl)C=CC=C2)=O)=O
JZC-215	0	CCCNC(C1=C(C=NN1C)NC(C2=C(Cl)C=C(Cl)C=C2)=O)=O
JZC-216	0	CN1N=CC(NC(C2=CC=C(C=C2Cl)Cl)=O)=C1C(NCC3=CC=CO3)=O
JZC-217	0	CCCNC(C1=C(C=NN1C)NC(C2=CC(S(=O)(N(C)C)=O)=C(C=C2Cl)Cl)=O)=O
JZC-218	0	CCCNC(C1=C(C=NN1C)NC(C2=C(Cl)C=C(Cl)C(S(=O)(N3CCCC3)=O)=C2)=O)=O
JZC-219	0	CC(CNC(C1=C(C=NN1C)NC(C2=CC=C(COCC(F)(C(F)F)F)C=C2)=O)=O)C
JZC-220	0	CC1=C(C(NC2=C(NN=C2C(N)=O)C)=O)C=CC(Br)=C1
JZC-221	0	CC(C(NC1=C(NN=C1C(N)=O)C)=O)NC(C2=CC=C(Cl)C=C2)=O
JZC-222	0	CC1=C(NC(C2=CC=C(OCC3CC3)C=C2)=O)C(C(N)=O)=NN1
JZC-223	0	CC1=C(NC(C2=CC(Br)=CC(F)=C2)=O)C(C(N)=O)=NN1
JZC-224	0	CC1=C(NC(C2=CC(F)=C(F)C(F)=C2)=O)C(C(N)=O)=NN1
JZC-225	0	CN(C(C1=NN(C(C)=C1NC(C2=CC=C(C=C2)C)=O)C)=O)C
JZC-226	0	CN1N=CC(NC(C2=NNC=C2Cl)=O)=C1C(N3CCCCC3)=O
JZC-227	0	NS(C1=CC=C(C=C1)CNC2=NC(C3=CC=CC=N3)=CC=N2)(=O)=O
JZC-228	0	CCC(NC1=NC(C2=CC=CC=N2)=CC=N1)C3=CC=C(S(C)(=O)=O)C=C3
JZC-229	0	NS(C(C=C1)=CC=C1CCNC2=NC(C3=CC=CC=N3)=CC=N2)(=O)=O
JZC-230	1	C1(NC2=CC=CC=C2)=NC(C3=CC=CC=N3)=CC=N1
JZC-231	0	CN(CC1=NC=CN1CC2=CC=CC=C2)C3=NC(C4=CC=CC=N4)=CC=N3

Table 10 Compound dataset (sample) for machine learning study. (continue)

Compound ID	**Activity**	**SMILES**
JZC-232	0	CN(CC1=NC=CN1CC)C2=NC(C3=CC=C(C)N=C3C)=CC=N2
JZC-233	0	CC1=NC(C)=CC=C1C2=CC=NC(NCC(C)N3N=C(C)C=C3C)=N2
JZC-234	0	NC1=NC(C2=CN(CC3=C(Cl)C=CC=C3Cl)C(C)=N2)=CC=N1
JZC-235	0	FC1=C(C2=CN=C(C)N2C(C)C)N=C(NC3=CC=C(C(N4CC[C@@H](NC)C4)=O)C=C3)N=C1

Note. Activity value 1 means protective against cisplatin, while 0 means no protective effect against cisplatin.

The file location of the entire compound dataset:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/dataset_activity.csv

Appendix B

Table 11 Mol2vec vectors (sample) of the compound dataset.

Compound ID	**mol2vec-000**	**mol2vec-001**	**…**	**mol2vec-098**	**mol2vec-099**
JZC-1	1.052486	0.927305	…	-13.0191	-1.76772
JZC-2	2.713912	1.827421	…	-10.0308	-3.68559
JZC-3	2.427935	2.503767	…	-12.0065	-2.10662
JZC-4	0.373738	3.765923	…	-10.6032	-4.68433
JZC-5	3.125469	-1.49734	…	-13.3179	-3.88405
JZC-6	2.952922	1.107688	…	-16.2054	0.810853
JZC-7	-2.42199	3.831441	…	-9.51197	-0.07321
JZC-8	0.206047	3.962412	…	-11.315	-1.16715
JZC-9	4.127991	-0.74615	…	-23.1811	-4.11265
JZC-10	-1.59946	5.908677	…	-12.4169	-1.17412
JZC-11	1.882193	-0.33321	…	-14.0631	-3.70532
JZC-12	0.764841	6.804127	…	-10.9767	-4.007
JZC-13	1.849891	1.370996	…	-10.9871	-3.19645
JZC-14	1.529856	-1.55927	…	-18.1177	-3.92226
JZC-15	1.040933	5.873358	…	-10.168	-5.32595
JZC-16	-2.24769	3.658378	…	-11.0843	-1.34151
JZC-17	2.538042	9.659736	…	-15.4411	-5.96477
JZC-18	2.777096	5.215434	…	-10.8554	-3.26038
JZC-19	2.463235	1.940857	…	-11.4866	-3.28527
JZC-20	2.150861	5.490986	…	-11.6875	-5.51498
…	…	…	…	…	…
JZC-200	1.933628	-0.97394	…	-15.7592	1.102077
JZC-201	1.373614	-0.96338	…	-15.2114	1.583534
JZC-202	2.924865	-0.07886	…	-16.6264	0.85126
JZC-203	2.443275	0.066873	…	-15.9823	0.956465
JZC-204	1.504102	-0.41144	…	-16.7607	-1.51201
JZC-205	0.753028	-1.24318	…	-15.5327	-1.08268
JZC-206	0.251455	1.153448	…	-11.1942	-4.29976
JZC-207	2.07379	0.79079	…	-16.0968	-1.29861
JZC-208	4.07919	-0.05883	…	-17.5966	-6.22719
JZC-209	2.862992	-2.6815	…	-15.6815	-2.94689
JZC-210	3.941807	-0.89134	…	-16.1428	-3.27097
JZC-211	0.995919	-1.35669	…	-10.0646	-0.18837
JZC-212	0.543512	0.420206	…	-12.2332	-1.92081
JZC-213	0.441556	-2.1181	…	-10.813	-0.61485
JZC-214	1.962303	-1.28805	…	-12.0048	-2.78796

Table 11 Mol2vec vectors (sample) of the compound dataset. (continue)

Compound ID	**mol2vec-000**	**mol2vec-001**	**…**	**mol2vec-098**	**mol2vec-099**
JZC-215	1.804642	-0.24502	…	-11.6566	-2.70939
JZC-216	-0.96347	0.601184	…	-12.1091	-1.5048
JZC-217	3.851608	-1.00402	…	-17.2319	-4.68065
JZC-218	4.51144	-1.23726	…	-15.6319	-5.49331
JZC-219	0.584949	-1.96508	…	-18.2691	-3.36905
JZC-220	1.589696	-1.79965	…	-12.7536	0.939225
JZC-221	4.471495	-1.5167	…	-17.8351	0.622057
JZC-222	3.392971	0.383118	…	-14.7918	0.488642
JZC-223	0.523873	-1.97448	…	-10.9889	1.403259
JZC-224	0.405894	-2.65815	…	-11.3846	1.325038
JZC-225	1.608622	-1.57759	…	-12.2974	-0.12924
JZC-226	1.711285	-3.24568	…	-12.5501	0.818426
JZC-227	0.625473	0.9778	…	-10.5508	-3.32771
JZC-228	4.177768	0.695673	…	-13.2371	-6.28058
JZC-229	1.41407	0.945515	…	-10.8648	-3.6641
JZC-230	-0.80922	1.646897	…	-6.58457	-1.83906
JZC-231	0.373032	1.294561	…	-12.2126	-2.32776
JZC-232	3.273234	0.500488	…	-13.0074	-2.38105
JZC-233	2.781635	0.85577	…	-15.0331	-1.36858
JZC-234	-0.82061	1.95022	…	-8.89985	-0.25377
JZC-235	1.014053	-2.17394	…	-17.5744	-0.21666

Note. The file location of Mol2vec 100 vectors for the entire compound dataset:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/dataset_mol2vec.csv

Appendix C

Table 12 Molecular descriptors (sample) generated by Schrodinger.

Compound ID	**i_desc_Atom_Count**	**i_desc_Atoms_in_Ring_System**	**…**	**r_qp_mol_MW**	**r_qp_volume**
JZC-1	23	18	…	373.208	1009.651
JZC-2	23	15	…	395.23	991.289
JZC-3	22	18	…	356.178	901.643
JZC-4	20	18	…	327.18	851.594
JZC-5	32	23	…	451.585	1426.204
JZC-6	25	17	…	382.249	1103.723
JZC-7	26	21	…	345.363	1104.382
JZC-8	26	21	…	344.375	1110.223
JZC-9	27	15	…	370.453	1185.888
JZC-10	33	23	…	486.319	1356.429
JZC-11	26	17	…	371.456	1182.622
JZC-12	22	18	…	293.281	876.424
JZC-13	25	15	…	345.348	1079.764
JZC-14	37	27	…	506.6	1612.2
JZC-15	25	20	…	371.222	996.834
JZC-16	27	21	…	360.374	1144.456
JZC-17	28	20	…	381.406	1193.389
JZC-18	18	14	…	241.249	774.405
JZC-19	20	15	…	267.33	931.35
JZC-20	27	20	…	362.384	1062.471
…	…	…	…	…	…
JZC-200	26	17	…	361.375	1111.092
JZC-201	25	17	…	365.794	1062.436
JZC-202	35	23	…	555.247	1362.67
JZC-203	36	23	…	569.274	1435.35
JZC-204	27	17	…	403.267	1215.49
JZC-205	31	21	…	440.931	1418.291
JZC-206	30	11	…	426.438	1369.348
JZC-207	25	11	…	350.367	1126.244
JZC-208	24	11	…	348.831	1115.107
JZC-209	20	11	…	292.724	904.777
JZC-210	27	16	…	407.255	1200.235
JZC-211	24	17	…	346.816	1060.46
JZC-212	22	11	…	320.778	1016.928
JZC-213	23	11	…	355.223	1060.872

Table 12 Molecular descriptors (sample) generated by Schrodinger. (continue)

Compound ID	**i_desc_Atom_Count**	**i_desc_Atoms_in_Ring_System**	**…**	**r_qp_mol_MW**	**r_qp_volume**
JZC-214	26	16	…	393.229	1076.251
JZC-215	29	11	…	462.35	1278.278
JZC-216	31	16	…	488.388	1405.014
JZC-217	31	11	…	444.428	1331.312
JZC-218	20	11	…	337.175	906.961
JZC-219	24	11	…	349.776	1053.53
JZC-220	23	14	…	314.343	1032.161
JZC-221	20	11	…	341.139	856.693
JZC-222	21	11	…	298.224	834.279
JZC-223	22	11	…	300.36	1027.741
JZC-224	23	16	…	336.78	1007.089
JZC-225	24	18	…	341.387	1060.559
JZC-226	26	18	…	368.453	1199.643
JZC-227	25	18	…	355.414	1118.6
JZC-228	19	18	…	248.287	848.765
JZC-229	27	23	…	356.429	1177.296
JZC-230	24	17	…	322.412	1106.751
JZC-231	25	17	…	336.439	1158.086
JZC-232	22	17	…	334.207	990.418
JZC-233	32	22	…	437.519	1404.898

Note. The file location of molecular descriptors for the entire compound dataset:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/dataset_descriptors.csv

Appendix D

Figure of Y-Scramble Function for Machine Learning

```
def yScramble(dataset):
    # 1. Create a new object for dataset
    import numpy as np
    import deepchem as dc
    ds = dc.data.DiskDataset.from_numpy(np.ones((2,2)), np.ones((2,1)), verbose=False)
    # 2. Creat a temporary array for y
    y_tmp = dataset.y
    # 3. Shuffle the array in place
    import random
    random.shuffle(y_tmp)
    # 4. Reset the shard for dataset
    ds.set_shard(0, dataset.X, y_tmp, dataset.w, dataset.ids)
    # 5. Return the scrambled dataset
    return ds
```

Figure 48 Figure of y-scramble function for machine learning.

The file location of Python script for y-scramble function:

~atao/Desktop/test_folder/machine_learning_test/deepchem_test/zebrafish/O20_datasets_scramble.py

Appendix E

Equations and Derivations for Confusion Matrix

$$TPR = \frac{TP}{TP + FN}$$

Equation 1 Sensitivity, recall, hit rate, or true positive rate (TPR).

$$TNR = \frac{TN}{TN + FP}$$

Equation 2 Specificity, selectivity, or true negative rate (TNR).

$$PPV = \frac{TP}{TP + FP}$$

Equation 3 Precision or positive predictive value (PPV).

$$NPV = \frac{TN}{TN + FN}$$

Equation 4 Negative predictive value (NPV).

Equations and Derivations for Confusion Matrix

(continue)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 5 Accuracy (ACC).

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Equation 6 Matthews correlation coefficient (MCC).

Appendix F

The 3D Structures of *C. sordellii* Mcs1

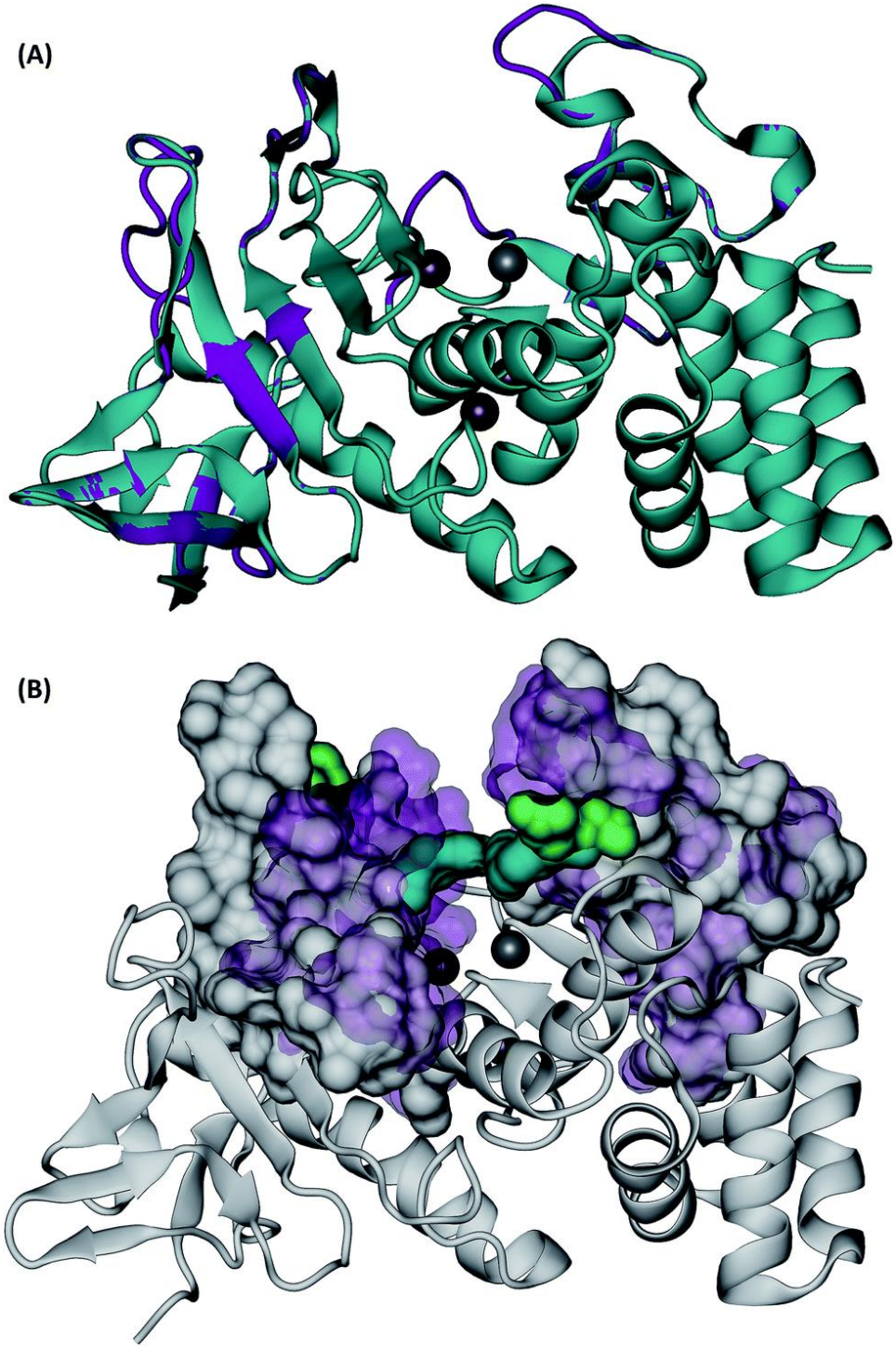


Figure 49 3D structures of C. sordellii *Mcs1 (A) Mcs1 aligned with* S. aureus *aureolysin.* C. sordellii *Mcs1 (purple),* Staphylococcus aureus *aureolysin crystal structure 1BQB (cyan), zinc ion (silver), calcium ions (brown). Homology modeling was performed using PRIME. (B) Superimposition of the most open and the most closed conformations of Mcs1 extracted from MD simulations. Open cleft conformation (Mcs1: white solid surface, Glu113-Arg227: green solid surface) and closed cleft conformation (Mcs1: purple transparent surface; Glu113-Arg227: cyan solid surface). In the closed cleft conformation, Glu113-Arg227 interactions (cyan) completely block access to Mcs1 zinc cleavage site.*

Appendix G

VCAM-1 Amino Acid Sequence Analysis

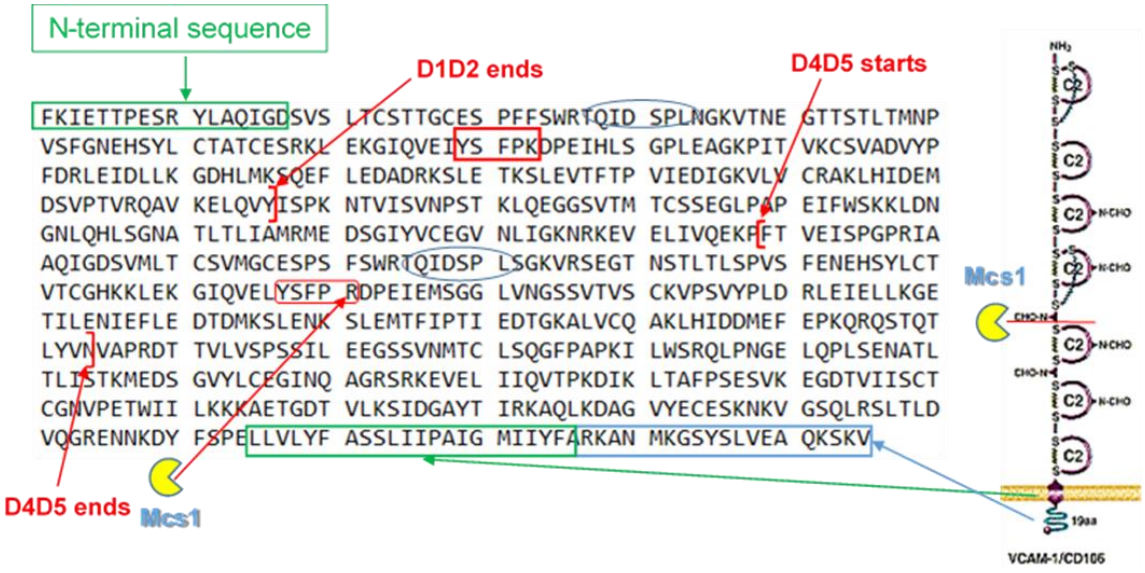


Figure 50 VCAM-1 amino acid sequence. The yellow Pac-man represents Mcs1. Mcs1 cleaves VCAM-1 at Arg381 of the D4D5 domain.

Appendix H

Transparent View of Mcs1 and VCAM-1 Docking Complex

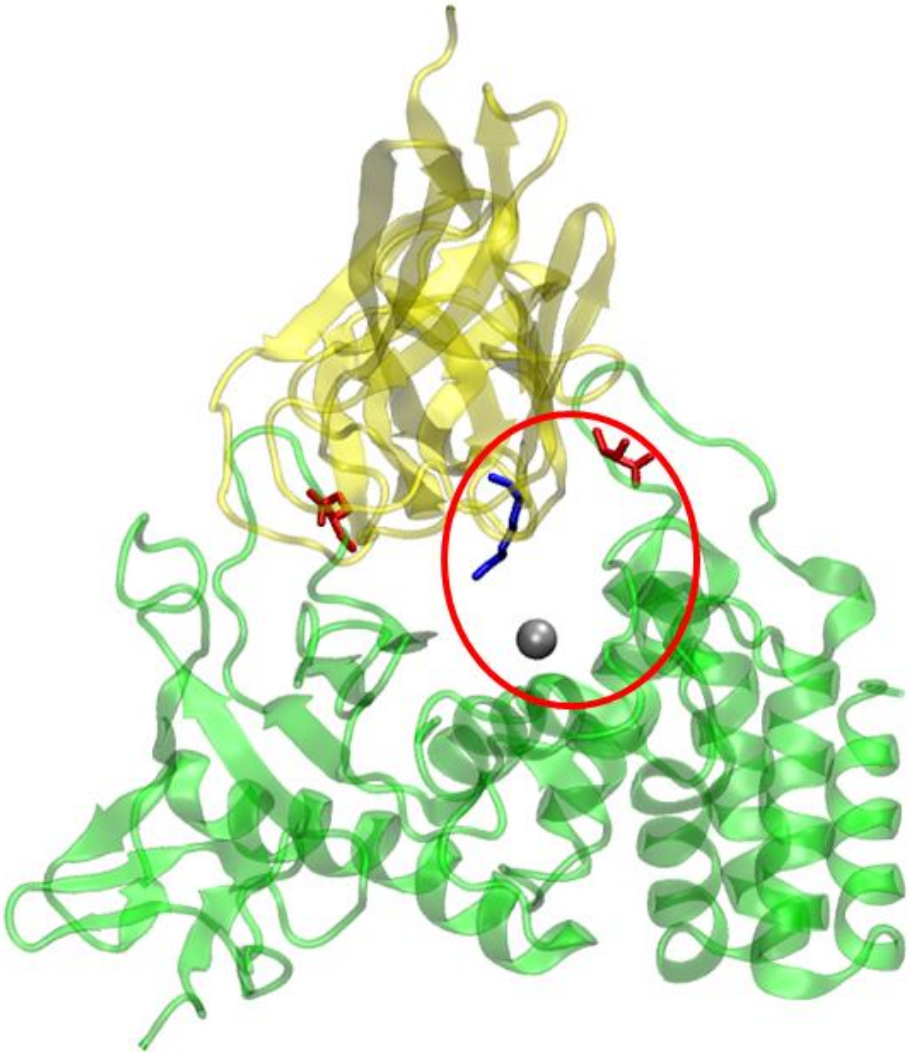


Figure 51 Transparent front view of the best docking result of Mcs1 and VCAM-1. Blue residue represents Arg381 of VCAM-1, red residues represent Glu113-Arg227 gate of Mcs1, and silver ball represents zinc ion of Mcs1.

Appendix I

The Electrostatic State of Mcs1 and VCAM-1 Docking Complex

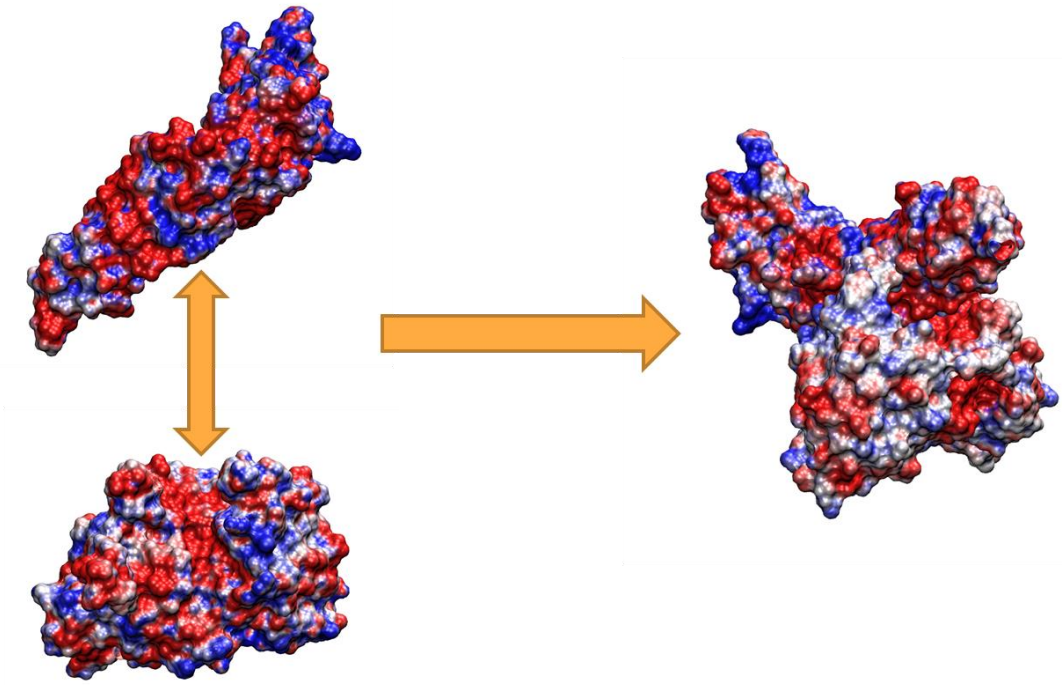


Figure 52 Electrostatic potential surface of Mcs1 and VCAM-1 docking complex.

Appendix J

Table 13 ROC-AUC of DUD benchmark test results.

	Morgan	3D	Glide HTVS	Glide SP	Vina	SMINA	23M Combo	3M Combo	2M Combo
ace	0.90	0.68	0.50	0.47	0.44	0.47	0.85	0.66	0.85
ache	0.79	0.75	0.64	0.61	0.67	0.65	0.78	0.75	0.78
ada	0.53	0.75	0.40	0.67	0.51	0.61	0.66	0.74	0.60
alr2	0.53	0.45	0.80	0.68	0.69	0.69	0.53	0.50	0.55
ampc	0.89	0.86	0.42	0.40	0.29	0.45	0.89	0.84	0.88
ar	0.72	0.83	0.90	0.86	0.78	0.78	0.84	0.85	0.81
cdk2	0.49	0.71	0.64	0.71	0.60	0.62	0.71	0.76	0.60
comt	0.87	0.28	0.71	0.67	0.52	0.56	0.65	0.33	0.76
cox1	0.58	0.57	0.84	0.73	0.72	0.63	0.57	0.56	0.59
cox2	0.90	0.95	0.91	0.92	0.77	0.89	0.95	0.95	0.94
dhfr	0.75	0.87	0.59	0.65	0.63	0.66	0.83	0.83	0.73
egfr	0.97	0.96	0.70	0.81	0.65	0.61	0.98	0.96	0.98
er_agonist	0.91	0.93	0.90	0.91	0.81	0.85	0.96	0.93	0.93
er_antagonist	0.88	0.96	0.90	0.73	0.73	0.72	0.94	0.95	0.90
fgfr1	0.58	0.59	0.55	0.50	0.44	0.43	0.57	0.55	0.53
fxa	0.75	0.38	0.72	0.75	0.61	0.64	0.59	0.40	0.65
gart	0.93	0.45	0.66	0.89	0.69	0.76	0.89	0.55	0.90
gpb	0.91	0.93	0.73	0.76	0.46	0.62	0.93	0.93	0.91
gr	0.63	0.74	0.78	0.73	0.49	0.50	0.63	0.65	0.56
hivpr	0.51	0.50	0.50	0.76	0.67	0.75	0.49	0.50	0.49
hivrt	0.59	0.66	0.81	0.75	0.64	0.65	0.67	0.69	0.67
hmga	0.95	0.93	0.89	0.84	0.48	0.86	0.96	0.97	0.97
hsp90	0.71	0.81	0.59	0.75	0.54	0.58	0.82	0.86	0.81
inha	0.72	0.73	0.59	0.63	0.48	0.52	0.71	0.71	0.70
mr	0.89	0.85	0.95	0.98	0.85	0.81	0.85	0.82	0.84
na	0.95	0.88	0.87	0.93	0.33	0.46	0.96	0.87	0.95
p38	0.27	0.54	0.52	0.62	0.58	0.57	0.36	0.58	0.35
parp	0.50	0.56	0.89	0.91	0.73	0.70	0.58	0.59	0.61
pde5	0.48	0.59	0.79	0.73	0.57	0.63	0.52	0.60	0.52
pdgfrb	0.50	0.35	0.35	0.28	0.29	0.30	0.45	0.33	0.48
pnp	0.99	0.87	0.83	0.90	0.37	0.56	0.96	0.85	0.94
ppar	0.96	0.77	0.51	0.79	0.72	0.81	0.98	0.74	0.97
pr	0.39	0.58	0.60	0.50	0.52	0.47	0.42	0.54	0.41
rxr	0.99	0.97	0.93	0.95	0.94	0.94	0.99	0.98	0.99
sahh	0.97	0.98	0.84	0.92	0.73	0.79	0.98	0.98	0.98

Table 13 ROC-AUC of DUD benchmark test results. (continue)

	Morgan	**3D**	**Glide HTVS**	**Glide SP**	**Vina**	**SMINA**	**23M Combo**	**3M Combo**	**2M Combo**
Thrombin	0.80	0.58	0.78	0.88	0.53	0.70	0.77	0.54	0.76
tk	0.89	0.84	0.69	0.90	0.59	0.63	0.89	0.85	0.89
trypsin	0.70	0.43	0.75	0.94	0.40	0.74	0.75	0.53	0.74
vegfr2	0.21	0.55	0.66	0.68	0.55	0.61	0.35	0.57	0.30
src	0.62	0.52	0.74	0.85	0.70	0.73	0.58	0.50	0.61

Appendix K

Table 14 Enrichment of DUD benchmark test results (Top 10).

	Morgan	3D	Glide HTVS	Glide SP	Vina	SMINA	23M Combo	3M Combo	2M Combo
ace	0.9	0.7	0.1	0.2	0.2	0.2	0.8	0.7	0.9
ache	1	1	0	0	0.3	0.1	1	0.9	1
ada	0.1	0.2	0	0.4	0	0.3	0.2	0.2	0.1
alr2	0.1	0.1	0.2	0.3	0.2	0.1	0.1	0.1	0.1
ampc	1	0.8	0	0	0	0	1	0.8	1
ar	0.8	0.9	0.8	0.6	1	0.9	0.9	0.9	0.8
cdk2	0.4	0.7	0.7	0.2	0.3	0.3	0.4	0.7	0.4
comt	0.1	0.2	0.2	0	0.2	0.3	0.3	0.2	0.3
cox1	0.6	0.5	0.4	0	0.3	0.1	0.6	0.6	0.6
cox2	1	1	0.9	0.8	0.7	0.8	1	1	1
dhfr	0.8	0.8	0	0	0.1	0	0.8	0.8	0.8
egfr	1	1	0.6	0.7	0	0	1	1	1
er_agonist	0.4	0.8	0.7	0.5	0.5	0.6	0.6	0.7	0.5
er_antagonist	0.5	0.3	0.4	0.2	0.4	0.6	0.6	0.3	0.6
fgfr1	0.5	0.4	0	0.9	0	0	0.6	0.3	0.5
fxa	0.5	0.6	0.3	0.6	0	0.1	0.5	0.5	0.4
gart	0.2	0.2	0.5	0.4	0	0	0.3	0.1	0.3
gpb	0.8	1	0.3	0	0	0	0.9	1	0.9
gr	1	0.8	0.6	0.5	0.7	0.6	0.8	0.8	0.8
hivpr	0.1	0	0.2	0.8	0	0.1	0.3	0.1	0.3
hivrt	0.7	0.6	0.7	0.6	0.3	0.3	0.8	0.6	0.7
hmga	1	1	1	1	0.2	0.1	1	1	1
hsp90	0.9	0.8	0	0.1	0	0	0.9	0.5	0.9
inha	1	1	0.6	1	0.6	0.8	1	1	1
mr	0.4	0.7	0.8	0.7	0.8	0.7	0.5	0.7	0.4
na	0.4	0.7	0.8	0.8	0	0	0.5	0.8	0.4
p38	1	0.8	0.1	0	0	0	1	0.9	1
parp	0.1	0.4	0.7	0.6	0.2	0.2	0.1	0.6	0.1
pde5	0.6	0.5	0.6	0.6	0.1	0.2	0.7	0.6	0.7
pdgfrb	1	0	0	0.8	0.9	0.9	1	0	1
pnp	0.7	0.7	0.3	0.1	0	0	0.9	0.6	0.8
ppar	1	0.2	0.1	0.7	0.2	0.2	1	0.2	1
pr	0.7	0.4	0.1	0.2	0.1	0.1	0.6	0.4	0.6
rxr	0.5	0.7	0.7	0.8	0.8	0.8	0.8	0.8	0.7
sahh	0.9	0.9	0.3	0.2	0.2	0.2	0.8	0.9	0.8

Table 14 Enrichment of DUD benchmark test results (Top 10). (continue)

	Morgan	3D	Glide HTVS	Glide SP	Vina	SMINA	23M Combo	3M Combo	2M Combo
thrombin	0.2	0	0.2	0.5	0	0	0.1	0	0.1
tk	0.2	0.4	0.1	0.1	0	0	0.4	0.4	0.3
trypsin	0	0.1	0.5	0.3	0	0.2	0	0.1	0
vegfr2	0.2	0.2	0.4	0.5	0.4	0.6	0.2	0.3	0.2
src	0.2	0	0.6	0.8	0	0.2	0.2	0	0.3

Appendix L

Table 15 Enrichment of DUD benchmark test results (Top 100).

	Morgan	3D	Glide HTVS	Glide SP	Vina	SMINA	23M Combo	3M Combo	2M Combo
ace	0.55	0.29	0.10	0.10	0.10	0.16	0.55	0.31	0.55
ache	0.62	0.39	0.08	0.05	0.17	0.12	0.61	0.39	0.62
ada	0.09	0.35	0.13	0.48	0.13	0.39	0.30	0.39	0.09
alr2	0.15	0.12	0.35	0.35	0.46	0.42	0.08	0.19	0.15
ampc	0.86	0.67	0.10	0.05	0.05	0.10	0.86	0.67	0.86
ar	0.39	0.45	0.39	0.30	0.42	0.43	0.42	0.43	0.38
cdk2	0.20	0.38	0.30	0.40	0.28	0.20	0.34	0.40	0.28
comt	0.82	0.18	0.36	0.45	0.27	0.36	0.45	0.27	0.73
cox1	0.40	0.32	0.48	0.44	0.52	0.52	0.32	0.36	0.36
cox2	0.98	0.98	0.78	0.72	0.47	0.70	0.98	0.98	0.98
dhfr	0.43	0.64	0.02	0.03	0.14	0.06	0.62	0.62	0.41
egfr	0.98	0.95	0.53	0.45	0.13	0.13	0.97	0.94	0.96
er_agonist	0.60	0.76	0.60	0.57	0.51	0.51	0.81	0.78	0.70
er_antagonist	0.46	0.79	0.54	0.56	0.18	0.36	0.79	0.87	0.59
fgfr1	0.18	0.12	0.02	0.19	0.01	0.00	0.19	0.12	0.19
fxa	0.11	0.06	0.21	0.40	0.02	0.07	0.09	0.07	0.12
gart	0.86	0.33	0.48	0.71	0.05	0.29	0.67	0.33	0.81
gpb	0.50	0.81	0.21	0.21	0.08	0.17	0.83	0.83	0.63
gr	0.17	0.21	0.15	0.09	0.10	0.12	0.26	0.21	0.22
hivpr	0.13	0.08	0.08	0.38	0.17	0.26	0.15	0.11	0.15
hivrt	0.20	0.25	0.40	0.40	0.20	0.18	0.28	0.30	0.23
hmga	0.94	0.80	0.66	0.60	0.06	0.43	0.94	0.86	0.94
hsp90	0.46	0.54	0.21	0.29	0.04	0.17	0.46	0.58	0.46
inha	0.54	0.46	0.15	0.29	0.24	0.28	0.54	0.40	0.54
mr	0.87	0.80	0.67	0.73	0.80	0.80	0.87	0.80	0.80
na	0.82	0.51	0.47	0.59	0.02	0.04	0.76	0.49	0.76
p38	0.30	0.26	0.12	0.01	0.07	0.07	0.32	0.24	0.30
parp	0.09	0.39	0.73	0.64	0.30	0.15	0.24	0.39	0.06
pde5	0.20	0.16	0.31	0.31	0.18	0.18	0.20	0.18	0.20
pdgfrb	0.19	0.00	0.01	0.10	0.11	0.11	0.13	0.00	0.15
pnp	1.00	0.84	0.28	0.52	0.00	0.04	0.96	0.68	0.84
ppar	0.90	0.14	0.02	0.33	0.11	0.32	0.85	0.15	0.84
pr	0.30	0.22	0.15	0.07	0.07	0.07	0.22	0.19	0.22
rxr	1.00	0.90	0.70	0.80	0.85	0.85	1.00	0.95	1.00
sahh	0.88	0.91	0.73	0.76	0.27	0.48	0.91	0.88	0.91

Table 15 Enrichment of DUD benchmark test results (Top 100). (continue)

	Morgan	3D	Glide HTVS	Glide SP	Vina	SMINA	23M Combo	3M Combo	2M Combo
Thrombin	0.42	0.05	0.26	0.55	0.00	0.22	0.22	0.03	0.28
tk	0.73	0.45	0.32	0.73	0.09	0.18	0.64	0.50	0.68
trypsin	0.09	0.07	0.43	0.52	0.00	0.30	0.05	0.07	0.23
vegfr2	0.04	0.12	0.26	0.27	0.12	0.16	0.07	0.15	0.11
src	0.24	0.03	0.25	0.53	0.07	0.14	0.21	0.02	0.20

Appendix M

Table 16 Mcs1 Docking Results (Top 40).

NSC	MVD			GScore			Emodel			Vina			Ranking Sum	Total Ranking
	Original	Open	Closed	Original	Open	Closed	Original	Open	Closed	Original	Open	Closed		
319990	3	14	4	14	61	1	5	7	1	5	4	5	124	1
80731	7	19	31	32	23	47	4	3	11	8	10	14	209	2
80735	4	30	10	132	34	72	9	4	4	19	7	8	333	3
319994	19	59	43	3	104	6	15	29	7	7	67	36	395	4
37553	32	6	11	54	225	23	8	14	17	3	2	3	398	5
654260	173	131	28	8	135	45	1	5	5	43	43	35	652	6
204232	29	12	41	272	198	20	19	13	27	24	11	22	688	7
227186	69	481	22	13	54	49	2	1	6	11	5	13	726	8
639174	28	71	107	22	315	43	14	46	12	99	73	25	855	9
60339	70	128	14	143	213	102	17	15	23	45	51	59	880	10
61610	2	4	1	94	712	58	7	11	2	1	1	2	895	11
5856	9	143	40	173	268	63	22	22	16	33	35	95	919	12
335979	15	53	8	149	43	83	39	23	10	189	231	110	953	13
67436	93	322	17	104	269	53	3	12	3	119	33	49	1077	14
177365	14	43	56	58	272	578	11	21	30	16	40	50	1189	15
320218	25	21	30	297	211	348	101	42	39	20	55	20	1209	16
134137	110	97	208	7	5	194	63	38	132	104	219	47	1224	17
341196	46	41	120	5	110	270	13	35	37	136	187	231	1231	18
73735	147	73	484	18	26	180	36	16	70	102	32	68	1252	19
71881	135	179	96	127	112	96	121	155	56	57	84	71	1289	20
202386	64	34	29	268	326	471	18	9	9	12	19	92	1351	21
91529	16	500	9	1	9	725	6	2	14	29	56	7	1374	22
122819	1	9	3	62	353	763	10	44	31	34	50	16	1376	23
133075	178	168	64	100	356	192	47	63	36	55	143	86	1488	24
146771	18	1	26	340	540	419	29	28	25	6	25	34	1491	25
46385	61	16	21	141	307	66	43	60	24	353	323	76	1491	26
81750	119	194	36	17	8	299	78	41	140	225	235	142	1534	27
143491	209	231	13	196	403	361	16	33	20	30	49	1	1562	28
280594	12	47	16	11	136	674	21	107	38	283	203	66	1614	29
311153	38	29	89	342	367	375	115	94	79	58	17	23	1626	30
146554	232	232	505	34	105	92	33	19	64	85	299	89	1789	31
58904	299	158	77	503	333	56	27	40	19	44	83	152	1791	32
63680	97	215	137	193	443	357	53	54	28	46	119	72	1814	33
164435	118	115	333	90	146	382	105	59	72	87	216	183	1906	34
329065	34	77	58	293	639	540	44	96	91	9	31	42	1954	35
50650	273	180	34	409	347	44	97	308	34	120	103	38	1987	36

Table 16 Mcs1 Docking Results (Top 40). (continue)

NSC	MVD			GScore			Emodel			Vina			Ranking Sum	Total Ranking
	Original	Open	Closed	Original	Open	Closed	Original	Open	Closed	Original	Open	Closed		
329255	121	296	114	76	108	229	81	70	65	166	348	348	2022	38
128606	114	88	203	712	89	505	147	30	111	25	12	41	2077	39
11668	56	23	6	443	258	557	214	123	69	140	153	45	2087	40
329249	259	257	349	95	175	313	82	68	54	56	180	109	1997	37

Appendix N

Figure of Mcs1 FRET Experimental Results (Top 40)

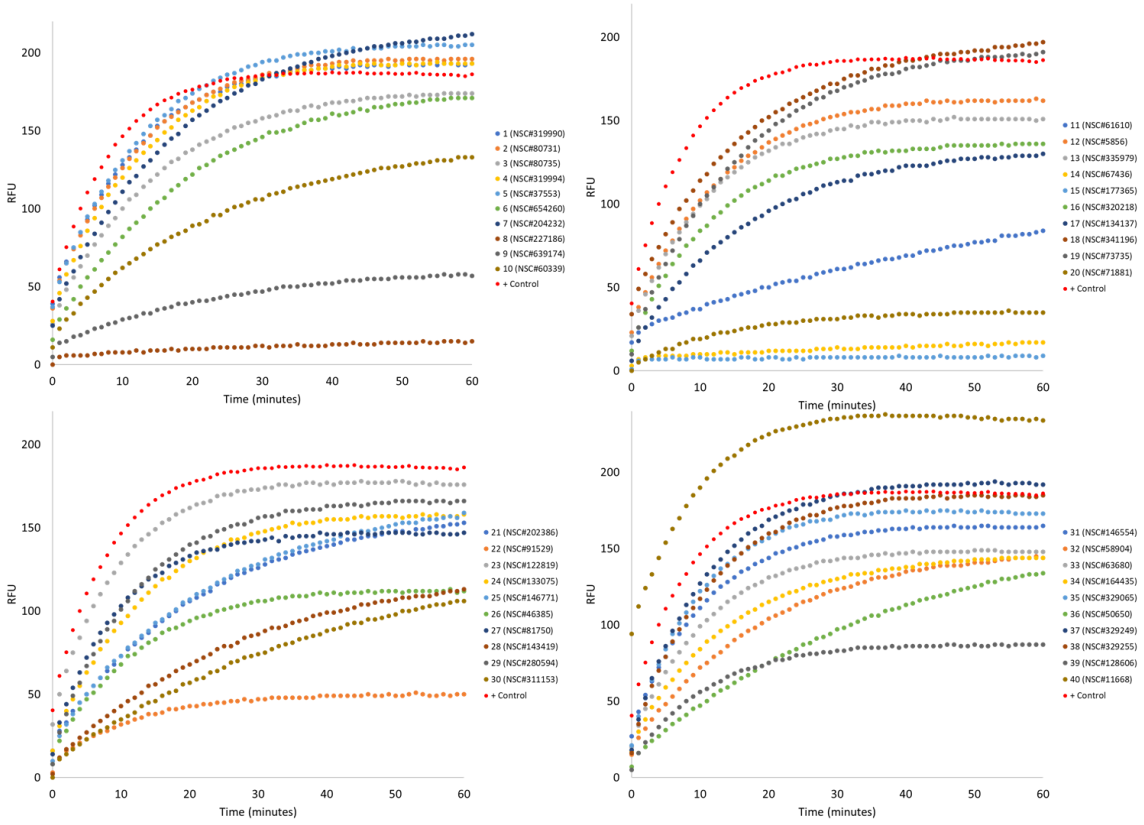


Figure 53 Mcs1 FRET Experimental Results (Top 40).