DOCKOMATIC: AN EMERGING RESOURCE TO MANAGE MOLECULAR DOCKING

by

Reed B. Jacob

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Interdisciplinary Studies

Boise State University

August 2012

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Reed B. Jacob

Thesis Title: DockoMatic: An Emerging Resource to Manage Molecular Docking

Date of Final Oral Examination: 18 April 2012

The following individuals read and discussed the thesis submitted by student Reed B. Jacob, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Owen M McDougal, Ph.D.	Chair, Supervisory Committee
Tim Andersen, Ph.D.	Member, Supervisory Committee
Julie Oxford, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Owen M. McDougal, Ph.D., Chair of the Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

DEDICATION

This thesis is dedicated to all those who have helped me in my journey. I would like to thank Dr. Owen McDougal for providing me the opportunity and direction for this thesis; my mother, Gena Jacob, for her support and encouragement; my father, Steve A. Jacob, for inspiring me, leading by example, and demonstrating the joys of science. I would especially like to thank my wife, Brooke Jacob, for her unwavering resolve to support me no matter what.

ACKNOWLEDGEMENTS

I wish to acknowledge Dr. Tim Andersen, Dr. C. Mark Maupin, Dr. Greg Hampikian, Dr. Julie Oxford, Casey Bullock, Dr. Ashley Fisher, and Ken Weekes for editorial comments and conversations that have strengthened the presentation of content described in this thesis.

Research funded by the Defense Threat Reduction Agency under contract number W81XWH-07-1-000, DNA Safeguard, NIH Grant #P20 RR0116454, the Idaho Idea Network of Biomedical Research Excellence, Research Corporation Cottrell College Scholars program, Mountain States Tumor Medical Research Institute, and Dr. Mark Rudin, Vice President for Research, Boise State University.

AUTOBIOGRAPHICAL SKETCH OF AUTHOR

Reed Jacob began his university education in 1999 and graduated with a B.A. from Boise State University in Music Composition. He then spent a few years exploring the continental United States. When his father, Steve A. Jacob, became ill with the diagnosis of stage IV pancreatic cancer, Reed returned home. He brought with him his lovely wife Brooke Jacob, and with her support returned to school at BSU. His journey into the academic world was marked by a shift from music to science. Realizing a tremendous aptitude for scientific discovery and computer-aided molecular docking, Reed entered the IDS program. Within three years, Reed has presented at a number of prestigious scientific conferences, contributed to major software development projects, and contributed to the publication of five manuscripts to date.

ABSTRACT

The application of computational modeling to rationally design drugs and characterize macro-biomolecular receptors has proven increasingly useful due to the accessibility of computing clusters and clouds. AutoDock is a well-known and powerful software program used to model ligand to receptor binding interactions. A limitation of AutoDock is the inability of a user to automatically create ligands and manage the input and output of data when dealing with large numbers of simulations; a problem that arises in High Throughput Virtual Screening (HTVS) or Inverse Virtual Screening (IVS). We have designed DockoMatic, a user friendly Graphical User Interface (GUI) application that constructs peptide-based ligands, integrates with the software program TreePack to create user defined peptide analogs, and automates the creation and management of AutoDock jobs for HTVS of ligand to receptor interactions. DockoMatic is a valuable tool for studying complex systems such as conotoxins, from the genus *Conus*, and their interactions with the well-characterized molecular receptor, *Aplysia californica* acetylcholine binding protein (*Ac*-AChBP).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

a.a.	amino acid
α-Ctxs	α-Conotoxins
Ac	*Aplysia californica*
AChBP	Acetylcholine Binding Protein
ADT	AutoDock Tools
AI	Aromatase Inhibitors
BCRP	Breast Cancer Resistant Protein
BDT	Blind Docking Tester
CADD	Computer-Aided Drafting and Design
CPU	Central Processing Unit
DOVIS	DOcking-based VIrtual Screening
DPF	Docking Parameter File
DLG	Docking LoG
ePMV	embedded Python Molecular Viewer
GPF	Grid Parameter File
GUI	Graphical User Interface

ΔG	Gibbs free energy of binding
HTVS	High Throughput Virtual Screening
IVS	Inverse Virtual Screening
K_i	Inhibition constant
MX	mitoxantron
nAChR	nicotinic Acetylcholine Receptor
NMR	Nuclear Magnetic Resonance
OSM	OncoStatin M
PDB	Protein Data Bank
PDBQT	Protein Data Bank with Charge(Q) and Torsions(T)
PyRx	Python Prescription
RCSB	Research Collaboratory for Structural Bioinformatics
SN-38	7-Ethyl-10-hydroxycamptothecin
THP	three-Huang Powder
VSDocker	Virtual Screening Docker

CHAPTER ONE: INTRODUCTION TO HTVS AND MOLECULAR DOCKING PROGRAMS

Introduction

Technological advances over the past 20 years have made it economically feasible to use computationally intensive algorithms for High Throughput Virtual Screening (HTVS) and Inverse Virtual Screening (IVS) of molecular interactions. HTVS involves docking many ligands to one or few receptors, while IVS docks many receptors to one or a few ligands. The sheer volume of chemical data has necessitated the emergence of computer programs for predicting molecular interactions between ligands and receptors, a process termed molecular docking. For drug discovery, the ligand may be a drug molecule and the receptor a protein with a structure that has been deposited in the Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB) [1]. Computational prediction of binding interactions can dramatically accelerate drug screening against biological receptor targets significant to disease treatment at a fraction of the expense of traditional methods.

Molecular docking programs, or "docking engines," are designed to accomplish two simultaneous tasks: 1) to identify the optimal binding orientation for a ligand within the binding cavity of the receptor, and 2) to score the resulting ligand binding interaction, providing a rank order that ideally predicts experimental results. Docking engines, such as DOCK [2] and AutoDock [3,4], calculate the optimal ligand binding orientation by minimizing the energy of interaction between molecules. Molecular docking results are

evaluated by visual inspection of ligand pose or quantitatively using a scoring algorithm. Scoring algorithms may be incorporated into the docking engine, or accessed through third-party software, such as XScore and Medusa Score [5,6]. Both XScore and Medusa Score have been shown to improve binding energy rankings over AutoDock when evaluated against a database of PDB benchmark standards. XScore is frequently cited as being used to re-rank AutoDock output and serves as the basis for AutoDock Vina [7,8,9,10].

DOCK and AutoDock were initially created during an era when computational resources for HTVS were prohibitively expensive and relatively primitive, but these programs have evolved over the years to be more user friendly, adaptable for HTVS, and useful as teaching and learning tools in a classroom setting. One noteworthy advance to AutoDock is a set of python scripts and programs called MGLTools that facilitate and automate workflow required for management of many simultaneous docking calculations. MGLTools contain a Computer-aided drafting and design (CADD) pipeline capable of accessing cloud resources for HTVS [11]. To enhance usability of DOCK and AutoDock, researchers have also developed Graphical User Interfaces (GUI) that automated job management and submission for molecular docking calculation. The focus of this chapter is HTVS GUI applications capable of processing large numbers of molecular docking calculations at an acceptable speed and cost, with reliable results, on a variety of computer platforms.

Docking engines calculate the Gibbs free energy of binding (ΔG) between a ligand and a receptor, which is fundamental to the understanding of complex systems in biochemistry and molecular biology. The calculation of ΔG is based on estimates of the

total energy of intermolecular forces of attraction including van der Waals interactions, hydrogen bonding, and electrostatic interactions. Ligands are ranked by the calculated ΔG value; lower ΔG values correspond to more favorable ligand binding, where higher ΔG values are less favorable (See Figure 1.1).

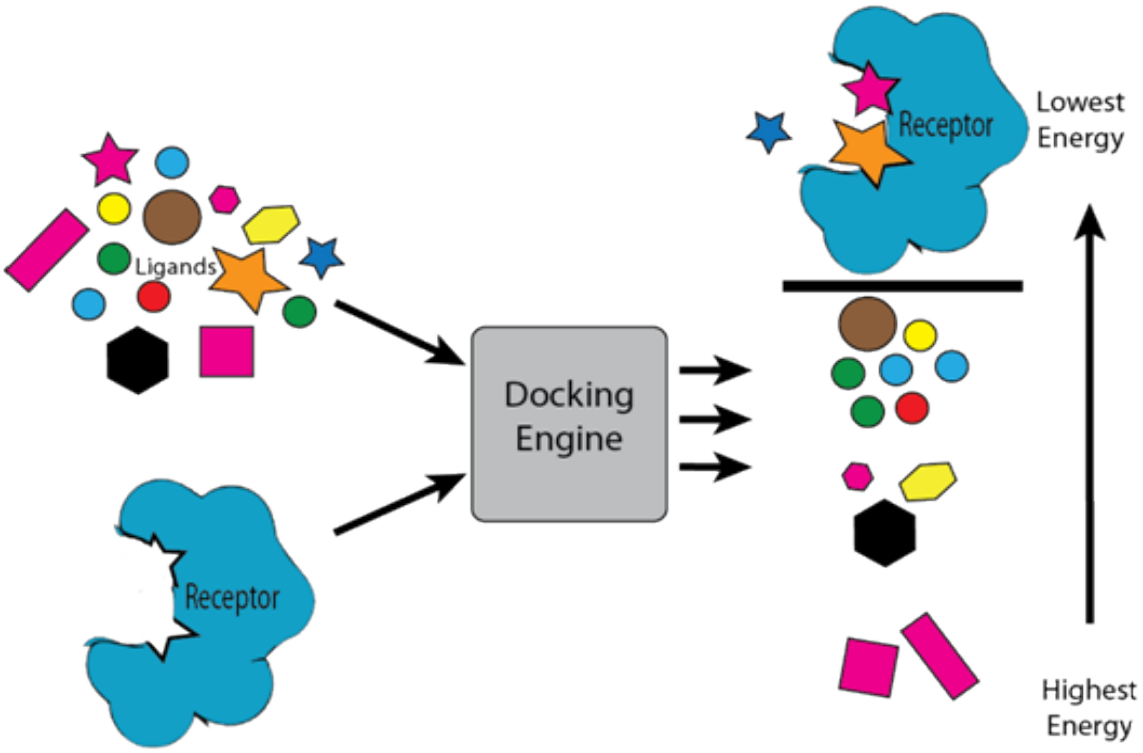

Figure 1.1 Depiction of high throughput virtual screening: multiple ligands are docked to a receptor and ranked by energy estimate.

Molecular docking experiments involving either DOCK or AutoDock require an inordinate amount of time to setup, submit, compute, and analyze results. HTVS programs solve these problems through process automation. HTVS programs that use DOCK and AutoDock as their docking engines include: DOVIS, VSDocker, WinDock, BDT, DockoMatic, PyRx, DockingServer, and MOLA. These HTVS programs are free

or inexpensive, and can run on hardware ranging from a personal computer to a computing cluster. A computing cluster typically consists of two or more computer nodes connected in parallel, with the ability to equally distribute computational jobs equally over the nodes. This exponentially increases computational capability. Cluster-based HTVS programs are DOcking-based VIrtual Screening (DOVIS) and Virtual Screening Docker (VSDocker), while WinDock and Blind Docking Tester (BDT) enable job queuing on only a single workstation. DockoMatic and Python Prescription (PyRx) can manage jobs independent of computer architecture, using a single workstation or cluster. DockingServer is a web-based application that operates regardless of user operating system, while MOLA works on networks consisting of homogeneous or heterogeneous computer architectures.

Researchers may select a molecular docking program best suited to their computing capabilities. Open access databases of receptor and ligand structures enable customized systems to be incorporated. Programs detailed here were selected, in-part, based on their use in solving research problems and their relative ease of use.

HTVS Program Requirements

Cluster Computing

DOVIS and VSDocker: DOVIS and VSDocker are comprehensive HTVS programs that automate and provide supporting features to AutoDock. These programs can manage millions of docking calculations on large computing clusters, and efficiently identify and order the top scoring ligands [7,8,9]. DOVIS is Linux-based, whereas VSDocker operates on Windows™. Both programs rank and score results via user-

specified criteria. DOVIS contains a plug-in for third-party scoring, such as XScore or Medusa Score [5,6].

DOVIS has been used to screen hundreds of RNA aptamers for binding to gentamicin [12]. Aptamers are single-stranded RNA or DNA molecules, generally around 50 base pairs in length. Aptamers bind specific small ligands, such as amino-sugars, flavin, or peptides, and are significant as diagnostic molecules associated with gene regulation. DOVIS 2.0 is an open source program under the GNU General Public License that is available for free download [13].

VSDocker is designed to manage jobs using Windows XP or 2003 servers. VSDocker matches DOVIS in speed and performance, based on an evaluation of molecular docking using ligands obtained from the ZINC database; run times were calculated to be 420 ligands/CPU/day [9,14,15]. VSDocker is free for non-commercial use but is not open source [9].

Standalone Computer Systems

WinDock: WinDock runs on a single Windows™ workstation. The docking engine for WinDock is DOCK. WinDock supports receptor homology model creation. Templates for receptors are identified via sequence alignment using ClustalX and T-coffee [16,17]. WinDock then directs Modeller to construct a homology model [18]. WinDock includes a large 3D ligand library, or the user can access compounds of interest from their own ligand PDB database. Users select force field, empirical, or knowledge-based ligand scoring algorithms to assess results [19-23].

WinDock has been used to study HIV-1 integrase enzyme binding to ligands isolated from three-Huang powder (THP), a Chinese medicinal formula [24]. Baicalein is

one of approximately 16 components in THP; baicalein was shown to inhibit infectivity and replication of HIV by agonizing HIV-1 integrase. HIV-1 integrase consists of three domains: N-terminus, core and C-terminus. WinDock identified the binding preference for baicalein to the middle of the ligand binding domain, the same site that was identified by co-crystallization with the inhibitor 5-CITEP [25]. A WinDock executable is available free of charge to students, academic instructors, and researchers by contacting the original author; the source code is not available [26].

BDT: BDT is a Linux-based HTVS application that uses AutoDock to automate blind docking, inverse virtual screening, and ensemble docking studies [27]. BDT was used to study the binding of volatile anesthetic ligands, like halothane or sevoflurane, to amphiphilic pockets in volatile anesthetic binding proteins like serum albumin and apoferritin [28]. BDT was used to predict that Van der Waals forces were the predominant factor in the binding of volatile anesthetic ligands to compatible binding proteins. BDT is free for academic and non-commercial research purposes, though not open source [27,28].

Standalone or Cluster Computing

DockoMatic: DockoMatic is a Linux-based HTVS program created at Boise State University that uses a combination of front- and back-end processing tools for file preparation, result parsing, and data analysis [29]. DockoMatic can dock secondary ligands and may be used to perform IVS [29,30]. The DockoMatic GUI facilitates job creation, docking, and result analysis for beginning and advanced users. The program can manage jobs on a single central processing unit (CPU) or cluster, and generates

ligand structure files by point mutation to an existing ligand PDB file or by entry of the single letter amino acid code for the peptide ligand sequence of interest.

DockoMatic has been used to study conotoxin binding to acetylcholine binding proteins (AChBPs) to investigate ligand binding determinants. AChBPs have similar homology to neuronal nicotinic acetylcholine receptors (nAChRs), which are pentameric ion channels responsible for the regulation of ions and small molecular neurotransmitters through biological membranes [31]. *Conus* snail venom peptides, specifically α-conotoxins (α-Ctxs), show targeted binding to both AChBPs and nAChRs (see Figure 1.2). As a step to evaluate conotoxin binding nAChRs, a study was performed that looked at crystal structures of α-Ctx's bound to multiple species of AChBPs. Conotoxin ligands that contained a public domain nuclear magnetic resonance (NMR) solution structure PDB file were analyzed in the bound state in the crystal structure, the peptide was removed from the ligand binding domain, and DockoMatic was used to redock the peptides. The peptides bound to AChBP included ImI[R11E], ImI[R7L], ImI[D5N], and PnIA[A10L:D14K]. The results demonstrated that DockoMatic may be used for computational prediction of peptide analog binding [29,30]. DockoMatic is free, and open source, for academic and non-profit use and available at http://sourceforge.net/projects/dockomatic/.

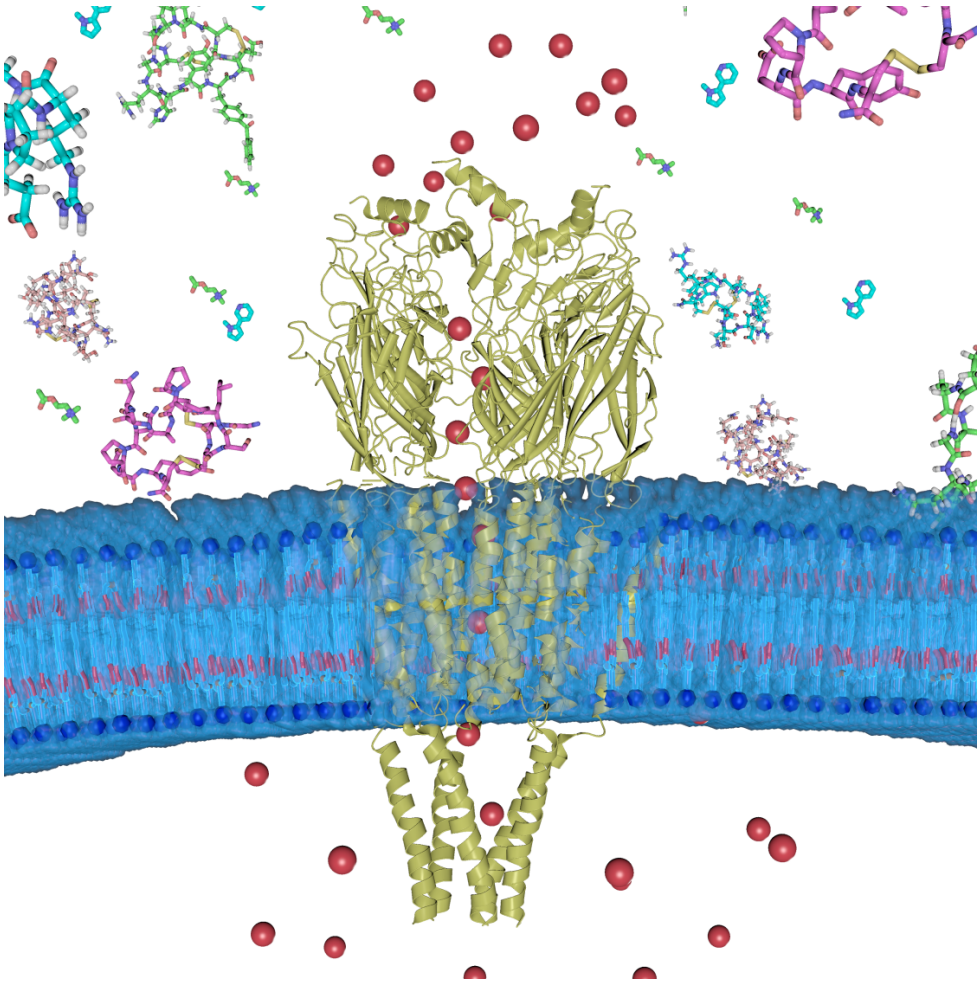

Figure 1.2 Depiction of α3β2 nAChR in a cell membrane surrounded by both agonist ligands and α-Ctx antagonists.

PyRx: PyRx runs on Windows™, Mac OS X™, Unix, or Linux computer clusters. PyRx can queue AutoDock jobs locally, or on a cloud using the Opal Web Services Toolkit [32,33]. PyRx includes an embedded Python Molecular Viewer (ePMV) for visual analysis of results, as well as a built-in SQLite database for result storage [34].

PyRx has been used to study aromatase inhibitors (AI). In post-menopausal women with breast cancer, increased levels of estrogen produced by the breast cancer cells increased cell production, creating a self-feedback loop [35,36]. AIs have

therapeutic value for patients that suffer from breast cancer associated with excessive aromatase activity [35]. The AIs studied using PyRx had known crystal structures; PyRx output was compared to X-ray structures to validate computational binding prediction [35]. PyRx is free, open source and distributed under the Simplified BSD license, and can be obtained from http://pyrx.sourceforge.net/downloads.

Hardware Independent

DockingServer: DockingServer is a comprehensive web service designed to make molecular docking accessible to all levels of users. DockingServer adds a MOZYME function, which uses atomic orbitals to calculate atomic charges, to its docking engine, AutoDock [37,38]. The process for job submission is straightforward, and the output report gives the specific bond type interactions between each ranked result and the target receptor. A drawback is that the docking output structure files are large and DockingServer user storage space is limited. Thus, the number of parallel processes that can be run, prior to transferring or deleting files, is restricted.

DockingServer has been used to investigate human breast cancer resistance using a homology model of breast cancer resistant protein (BCRP) to characterize the potential interaction modes of the substrates mitoxantrone (MX), prazosin, Hoechst33342, and 7-Ethyl-10-hydroxycamptothecin (SN-38). Results indicated there is a central cavity in the middle of the lipid bilayer of BCRP capable of containing two substrates, instead of the previously hypothesized single substrate [39]. This study illustrates a possible mechanism for BCRP function that may lead to inhibitors for future drug development. The DockingServer web-based service is available for a modest annual subscription.

MOLA: MOLA runs off a CD boot disk that preempts the local operating system with its own operating system [40]. MOLA is capable of configuring a temporary computer cluster from heterogeneous, networked standalone computers regardless of operating platform. This program is intended for research labs without access to a dedicated computer cluster. MOLA includes AutoDock Tools (ADT), which is a program included within MGLTools, for GPF (Grid Parameter File) creation and ligand/receptor preparation. ADT also generates an analysis spreadsheet ranked by the lowest binding energy and distance to the active site [11]. MOLA does require some familiarity with ADT and preparation of receptor files for AutoDock submission.

MOLA was used to investigate ligand binding to Retinol binding protein, HIV-1 protease and Trypsin-benzamide, each with a ligand library search of over 500 ligands and decoys, recreating the approximate potential bell curve of these ligand sets to each receptor. MOLA is a free download as an image file for direct burning to disk [40]. The source code is not available.

Summary

The role of computational molecular docking in educational, research, and drug discovery is evolving at a rapid rate. Access to this field by an ever increasing number of students, teachers, and scientists has been facilitated by software programs similar to those described here. Each program we describe has been used to address real-world research problems that demonstrate the potential benefits of molecular docking in many fields of study. Table 1.1 summarizes the capabilities and attributes of each HTVS program reviewed. Individuals should select a program to use dependent upon their computer hardware access, financial resources, and desired objectives.

Table 1.1 A review of HTVS programs with GUIs available to educators.

	WinDock	BDT	Dovis	VS Docker	DockoMatic	Docking Server	PyRx	MOLA
PlatForm	Windows	Linux	Linux	Windows	Linux	Web	Linux, Unix, Windows, Mac OS X	All
Release Date	2007	2006	2008	2010	2010	2009	2009	2010
Reference	[26]	[27]	[7,8]	[9]	[29,30]	[37,38]	[32]	[40]
Homology Modeling	√							
Ligand Library	√	√	√	√	√	√	√	√
Ligand Creation					√	√		
Open Source			√	√	√		√	
Cluster/Cloud			√	√	√		√	√
Installer*	√			√		N/A	√	√
Local Resource Demand$	S	S	M	M	E	S	E	E
Documentation&	1	1	2	2	1	5	3	2
Ease of Use#	1	1	3	3	2	1	3	4

* if an N/A appears that program needs no installer; it is a web interface.
$ S – minimum program requirement is a single computer workstation; M – multiple computers in a cluster are required, and E – single or multi-processor enabled.
& Rated on a 1-5 scale with 1 being basic installation instructions to 5 being in depth tutorials and worked examples for applications.
# Rated on a 1-5 operator scale with 1 being a user with basic computer skills to 5 being an experienced programmer.

The HTVS programs described here were developed with the common goal of enhancing the ability to perform molecular docking studies using one of two well-established docking engines, DOCK or AutoDock. The optimal program for use in instruction or research is dependent on the specific goals and needs of the project. For a researcher in a department with limited computer availability interested in occasional docking investigations, we suggest WinDock or PyRx, as both programs are available for a Windows™ operating system. For more in-depth docking studies with Linux operating system availability, BDT, PyRx, and DockoMatic may be preferable. If a Linux cluster is available, then DockoMatic, DOVIS, or PyRx are recommended, or VSDocker for a Windows™ cluster. If there are multiple networked computers, without a cluster, MOLA is ideal for HTVS. For those with limited computer resources, DockingServer is an

external web service for a reasonable subscription. Of these programs, DOVIS, VSDocker, and BDT provide rank ordered lists of results, with limited capacity for the user to visualize the docked molecules without accessing another software program like PyMol. For result visualization, DockoMatic and MOLA provide a link directly to PyMol and ADT, respectively [41,42]. WinDock, PyRx, and DockingServer contain fully integrated visualization capabilities for all steps in the process of docking to result analysis.

In addition to computational requirements, each HTVS program has unique features to assist in docking studies and data analysis. BDT is optimal if the project-specific receptor does not have a known binding pocket. If homology model construction is required, WinDock contains a Modeller interface. If the primary goal is limited to screening ligands, then DOVIS or VSDocker work well. To study point mutations of small cyclic peptides like conotoxins or other peptide ligands, then DockoMatic with automated peptide analog structure creation is a recommended option. PyRx is useful for ligand comparison studies because it offers well-integrated storage and visualization of HTVS results that facilitate binding analysis. For those new to the field of computational chemistry, DockingServer is a comprehensive, user-friendly, and supported program.

The goal of all molecular docking studies is to increase understanding of the interaction between molecules, whether protein-protein, or protein-ligand. This broadened knowledge base can then serve to direct wet bench experimentation with minimal cost and labor.

CHAPTER TWO: DOCKOMATIC: DESIGN AND DEVELOPMENT

Background

Several computer programs have been developed to estimate the Gibbs free energy (ΔG) for molecular docking by calculating the energy associated with atomic interactions between the ligand and a target receptor [43,44]. Examples of more popular molecular docking programs include AutoDock [3,43], MOE-Dock [45], GOLD [46], DOCK [47], and Glide [48]. Of these, AutoDock is the most widely cited resource for simulating ligand docking to receptors [43]. AutoDock and other similar programs rank ligands based on ligand to receptor binding interaction energy estimates [49]. The strength of AutoDock is the computational algorithm, which uses a combination of linear regression analysis in concert with a genetic algorithm and the AMBER force field [50]. The AutoDock application works very well for the analysis of a single ligand with a specified receptor. However, AutoDock is not efficient at screening many peptide ligands binding to a protein receptor. In these high throughput virtual screening (HTVS) instances, it is necessary to run ligands individually through AutoDock, followed by manual analysis of the output file to assess ligand interaction results. This process is time consuming, both computationally and for the user. The work described in this chapter presents DockoMatic, a Graphical User Interface (GUI) application designed to facilitate the use of AutoDock for HTVS, by automating the setup, submission, and management of AutoDock jobs, and summarizing and easing analysis of results.

DockoMatic was developed in concert between Dr. Tim Andersen in the Department of Computer Science, and Dr. Owen McDougal in the Department of Chemistry and Biochemistry at Boise State University. Casey Bullock, a graduate student in Dr. Andersen's laboratory did much of the coding for DockoMatic, while I contributed to GUI design layout, program functionality algorithms, such as peptide ligand creation, peptide analog creation, ease of use, validity testing, algorithm creation for specialty applications, and use of the program for a variety of projects. Our initial goal was to develop a program that would simplify and speed the process of creating peptide ligands and simulating the docking of those ligands to biomolecular receptors. DockoMatic's intuitive user interface greatly reduces the amount of user time required to setup, submit, and analyze AutoDock jobs.

DockoMatic was created with the following major features:

- Intuitive GUI for user-controlled automation
- Create, Submit, and Manage AutoDock jobs
- Peptide-ligand creation based on single letter amino acid codes
- Peptide-analog structure generation from parent peptide structure file
- High Throughput Virtual Screening (HTVS)
- Summary, screening, and analysis of results by an intuitive process

While other tools are available to use AutoDock on clusters of computers [8], no tool that we are aware of includes all of the features of DockoMatic in a single package, and no tool at this time has automated the creation of peptide ligand structure files, nor creates automated analogs.

DockoMatic was designed to have a simple and intuitive interface for use by an advanced or novice scientist with limited computer science training. A set of tools, MGLTools, has been provided with AutoDock, which contain a GUI called AutoDock Tools (ADT) [11]. This ADT interface provides all the necessary tools to prepare and submit jobs to AutoDock. The difficulty with this interface is its use in HTVS studies. ADT requires a user to be familiar with all aspects of docking in order to effectively prepare a docking job. The workflow to use ADT is illustrated in Figure 2.1.

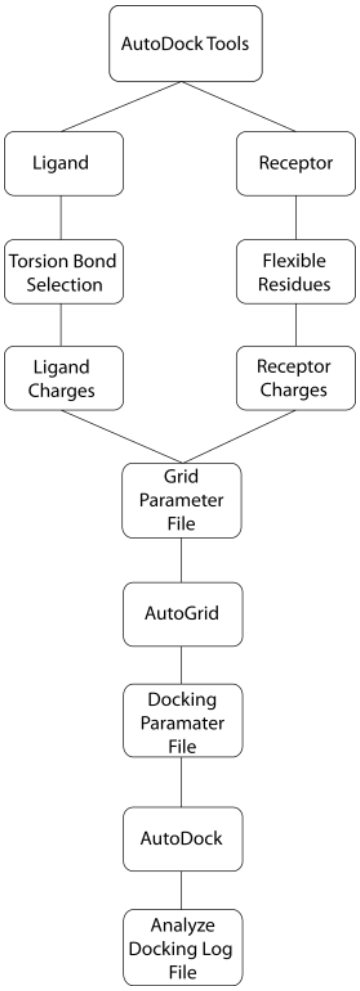

Figure 2.1 AutoDock Tools workflow for AutoDock job submission.

To manually submit a docking job for AutoDock, the ligand PDB file must be manipulated to select flexible torsion bonds, and add atomic charges. A similar process is needed for the receptor PDB file. Once both the receptor and ligand have been prepared, it is necessary for the user to create a grid parameter file (GPF), which is then manually submitted to AutoGrid. AutoGrid is a preprocessing tool provided with AutoDock that calculates the necessary energy maps for AutoDock to evaluate and estimate ΔG. After AutoGrid is complete, which takes an average of 10 minutes, the user must then create a docking parameter file (DPF) in order to run AutoDock. This file lists the generated maps for all ligand atom types, and both the ligand and receptor file names. This is the DPF that is submitted to run AutoDock. At the completion of AutoDock, which takes anywhere from a few minutes to many hours, depending on the system and size of the grid, the user must manually analyze the results either using ADT, which can read the AutoDock result files, or docking log (DLG) files. Alternatively, the user may manually extract each of the results into a separate PDB file. This process is incredibly time consuming, and is not conducive to HTVS experiments.

Features

Intuitive GUI for User Controlled Automation

It is the tedious and time consuming manual workflow that DockoMatic was developed to automate (see Figure 2.1). The user does not have to be knowledgeable in scripting or computer languages to efficiently perform functions in DockoMatic commonly associated with command-line driven programs. Instead, the interface was created to guide the user through the requirements for a successful AutoDock job

creation, submission, and result analysis. The first step in DockoMatic's design was to decide which files were necessary to successfully run an AutoDock job. After examining each AutoDock job creation step, and determining the underlying commands, it was decided that only a few files were required to be supplied by the user. These files are both ligand and receptor PDB files, the output directory, and a grid box in the GPF file format. To this end, DockoMatic's design layout places the user required items on the left, the job information or management grid in the center, and the program options on the right (see Figure 2.2) [29].

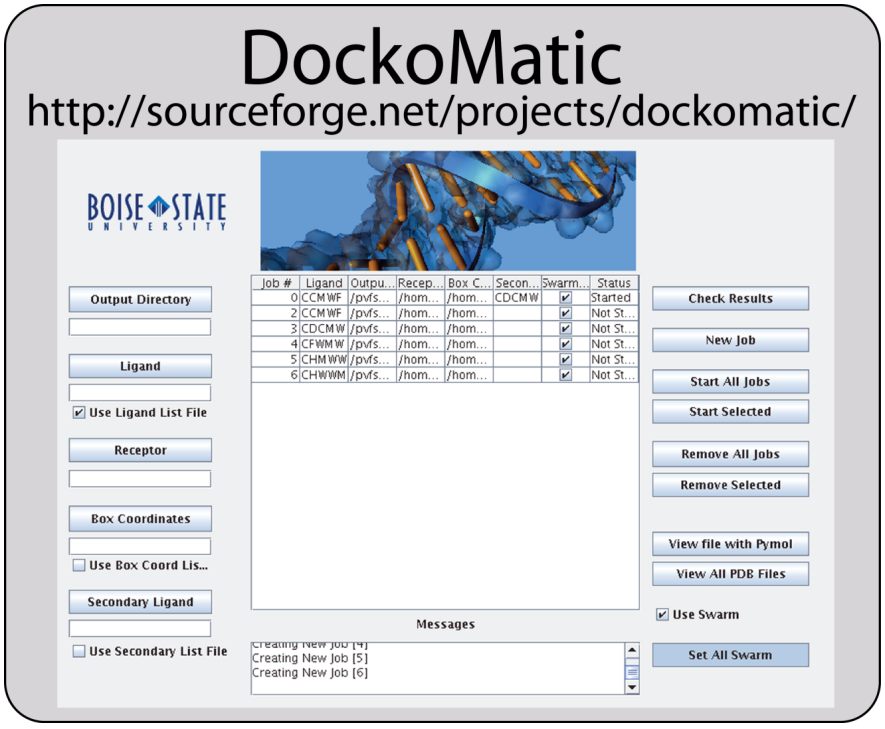

Figure 2.2 DockoMatic GUI interface. The Graphical User Interface for DockoMatic: user input fields (left), current processing status (center), and results/analysis fields (right).

The left side of the window detailing user input requirements begins with the output directory. The user selects this box and navigates to the directory where they want

DockoMatic to output the results. If no output directory is specified, the default is the directory where the user is when the GUI was started. The ligand box is where the user can either select a single PDB file, or input a string of amino acids. For HTVS, instead of entering an individual ligand, the user may enter a file name and check the box "Use Ligand List File." In this case, the file name must refer to an input file that contains either a list of single letter amino acid codes for each peptide to be created, or a directory path to existing ligand PDB files. In a similar manner, the user selects both the "Receptor" and "box coordinate files." Users may also choose to specify a secondary ligand or a file containing a list of secondary ligands to model how an additional ligand may bond in the presence of the first ligand [29].

For example, all that is needed for DockoMatic to successfully queue a basic AutoDock job is a ligand PDB file, a receptor PDB file, a GPF file, and a place to put the results or output directory. So to submit the file, the files could be named as follows, the ligand PDB file, "ligand.pdb," and the receptor file, "receptor.pdb," with the grid parameters being "receptor.gpf". The first step would be to select a directory for the output. Each of these files would be input into DockoMatic as described above. Once done, the user may create AutoDock jobs by pressing the "New Job" button. This populates the management grid with a list of all jobs. Since DockoMatic may be used to facilitate HTVS, the total number of jobs created is equal to the cross product of the ligands, receptors, and box coordinates. For example, if just one of each were provided, then the total number of jobs would be one. But if the user had a list of 10 ligands, with one receptor, and one box coordinate, then the total number of jobs would be 10. At this point, the job specifications can be manipulated before the jobs are started. If the job

details are satisfactory, selection of the “Start All Jobs” button will start all jobs. Figure 2.3 lists the files required by DockoMatic to perform one or many docking jobs. The basic work flow for DockoMatic allows the user to submit the three necessary files and an output directory followed by selection of the “New Job” button to populate the jobs. The “Start all jobs” button submits the jobs to AutoDock. At the completion of the AutoDock calculation, DockoMatic parses the results and creates a ranked list that can be readily analyzed by the user. If the user wishes to start an individual job, they can do so by selecting the desired job and pressing the “Start Selected” button. Jobs may be stopped and removed from the management grid with either the “Remove All Jobs” or “Remove Selected” buttons.

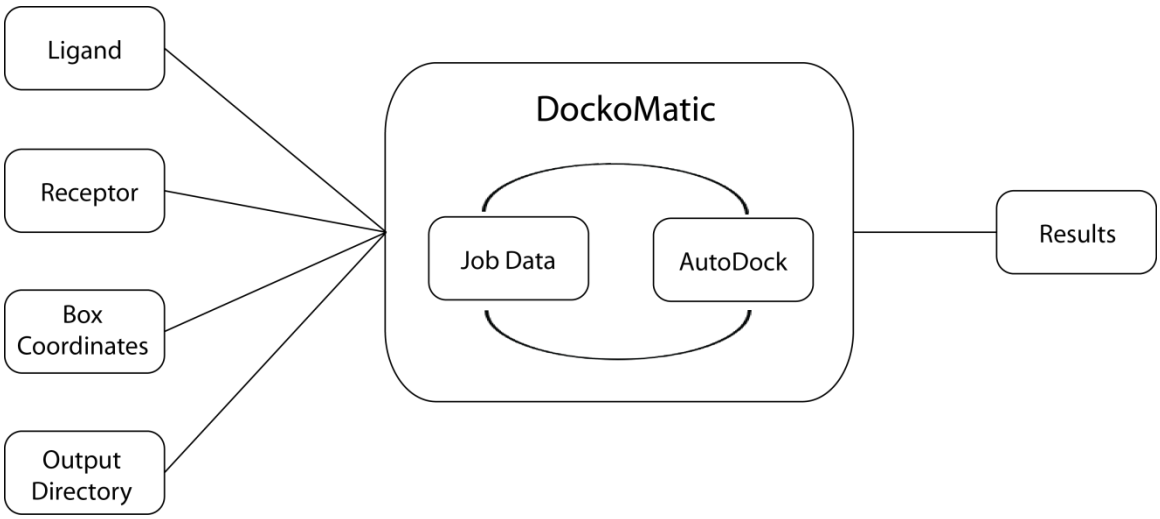

Figure 2.3 DockoMatic job workflow.

The management grid lists the job number, ligand specified or path to PDB file, output directory path, path to receptor, path to box coordinate file, secondary ligand or path to secondary ligand file, whether the job is a swarm job, and the current status of the job. Swarm can be specified, via a checkbox, for parallel job submission to a cluster, or

jobs can be spawned as individual processes on a single workstation. Once jobs are started, the status of each job in the window is automatically checked every ten seconds. Below the management grid is the "messages box" listing the progress of the submission.

As DockoMatic detects job completion, it extracts all result ligand conformations into individual PDB files and compiles a results file listing important information, such as the estimated ΔG and an estimated inhibition constant (K_i). These results can then be viewed by selecting the PyMol buttons to view individual results, or select a directory to view all the PDB files in that directory. This ability to view each job result using PyMol requires that PyMol be installed and accessible via the user's environmental variables [41].

Create, Submit, and Manage AutoDock jobs

While DockoMatic significantly reduces the time required by the user to create and submit jobs to AutoDock, there are a few files the user must provide. These include:

1) the ligand PDB file or sequence
2) a receptor PDB file
3) a user-defined template Grid Parameter File (GPF)

Of the required files, the GPF is the most difficult to generate. A typical GPF contains specific atom types as defined by the ligand and Cartesian box coordinates, specified by a single center point and directional dimensions, thus making each GPF ligand specific. It would be redundant to automate an HTVS process while requiring manual creation of ligand-specific GPFs. This would negate the automation. To prevent this, DockoMatic requires one GPF to use as a template to automate the creation of

ligand-specific GPFs based on the submitted HTVS ligand list. DockoMatic maintains the Cartesian box coordinates while adjusting the atoms and maps to be ligand specific.

AutoDock requires that both ligand and receptor PDB files undergo preparations as illustrated in Figure 2.1. This process results in new files with a file extension of PDBQT, where the PDB represents the standard protein data bank file extension, and the addition of the QT represents charge (Q) and torsion (T) bonds. These PDBQT files are then used for molecular docking calculations. AutoDock creators have provided MGLTools, which contain the necessary command line driven scripts and utilities for ligand and receptor file preparation [11]. DockoMatic calls the necessary conversion utilities, specifically: prepare_ligand4.py, prepare_receptor4.py, prepare_gpf4.py, and prepare_dpf4.py; establishing a pipeline to create the full GPF and the necessary DPF for AutoDock. By default, AutoDock runs 10 stochastic simulations per compound to find the best docking site for a ligand within the specified grid space on the receptor. Running 10 simulations provides a rapid screen of potential binding sites, but it has the disadvantage of returning less accurate results than longer runs of 50 to 100 simulations. AutoDock documentation recommends using a minimum of 50 docking simulations to ensure accurate molecular docking results with the added comment that more simulations will typically result in improved statistical results. Because of this, we have set DockoMatic to default to 100 AutoDock simulations, a number consistent with that reported by others in the literature [51]. In addition, we have modified the number of maximum energy evaluations from AutoDock's default of 2.5 million to be one million. After experimentation, this was found to be the minimum number to maintain result integrity, as well as increase speed and efficiency.

Peptide-Based Ligand Creation

AutoDock requires submission of coordinate files in PDB format for all ligands. This is not a problem if the PDB files exist, but if they do not, then the creation of novel ligand structure files can be time consuming and tedious. To manually create a peptide ligand requires a third-party software program like Spartan [52]. The user may manually select each residue, one at a time, in sequence to construct the ligand. The ligand creation procedure is time consuming, especially when performing HTVS studies with peptide ligands.

DockoMatic automates peptide-based ligand creation, either as a prelude to creating an AutoDock job, or as its primary function. DockoMatic constructs a PDB file for a ligand based on the user supplied string of alphabet characters representing the single letter amino acid sequence of the ligand. For example, if the user wanted to create the tripeptide Trp-Lys-Val, they would enter in the ligand box the letters WKV, and DockoMatic would create a PDB file for the ligand as illustrated in Figure 2.4. This is a time-saving measure that facilitates job setup. DockoMatic creates peptide ligands using pre-created PDB files. The algorithm to create a ligand structure from a peptide ligand string can be summarized as follows; code for this algorithm was written by Casey Bullock.

1. beginning (N-terminus)
2. if next amino acid is not proline, add backbone structure, else add proline
3. add amino acid side chain
4. repeat steps 2 and 3 until the ligand string is exhausted

5. add end (C-terminus)

6. optimize ligand structure

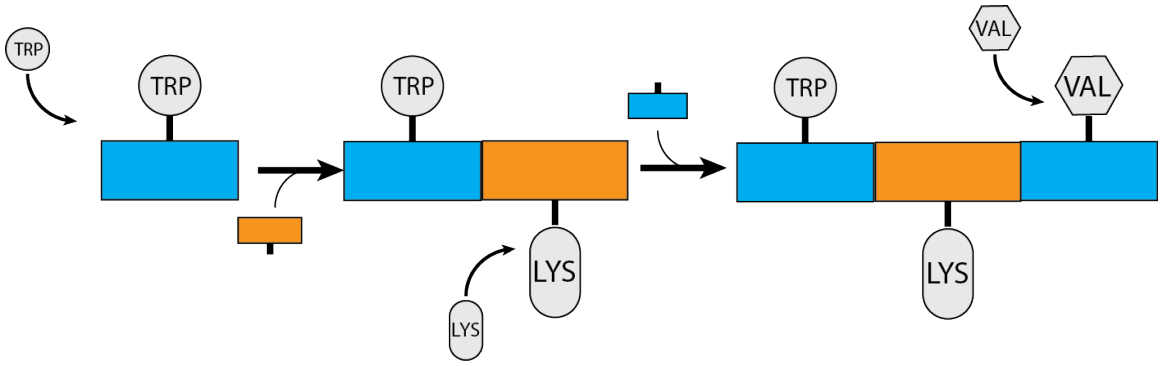

Figure 2.4 DockoMatic Peptide Linear Ligand Creation. Illustrates the process of creating the tripeptide WKV.

Proline was treated separately from the other amino acids due to the backbone bend associated with the presence of this amino acid in the structure of a peptide or protein. Since the backbone is built into the side chain PDB file, for proline, no additional backbone adjustment needs to be made when proline is encountered on the ligand. To avoid unintended atomic collisions, the orientation of side chains on sequential amino acids alternate up and down. In total, there are 44 PDB files used for ligand creation; one for the N-terminus and one for the C-terminus, a backbone with the side chain of the twenty common amino acids oriented up, and a backbone with the side chains oriented down. Following the complete formation of the PDB ligand structure, DockoMatic utilizes the computer program Obconformer, from the Open Babel package, as an energy optimization tool [53]. This feature was added because of our interest in HTVS of pentapeptide ligands. The current project involves the study of non-naturally occurring

pentapeptides that show interesting biological activity. DockoMatic was designed to be able to screen thousands of these peptides against potential targets for drug discovery.

Peptide-Analog Creation

The automated analog creation feature in DockoMatic provides an *in silico* method of site-directed mutagenesis for complex peptide and protein structures based on experimentally determined tertiary structure. To manually accomplish the same task requires the construction of a complete homology model, a process that entails rebuilding the structure from scratch using the existing ligand as a template. Homology model creation is a complex process because it requires knowledge of computer scripting to use Modeller, the most common program for this purpose [18]. DockoMatic automates this process, the peptide analog structure file creation utility enables combinatorial computational high throughput screening of peptide ligands against biological receptors. This feature was implemented due to our interest in the field of conotoxin research. Conotoxins are small 15-30 a.a. peptides constrained by a molecular scaffold where minor variation in primary sequence may cause major changes in peptide binding characteristics. To investigate this phenomenon computationally, a predictive method for computational simulation of peptide analogs was developed. To implement automated peptide analog creation into DockoMatic required incorporation of the command-line driven utility TreePack, a program to perform side chain replacement in the creation of peptide analogs [54,55]. TreePack is a software tool created for application in protein homology modeling; it is comparable to the widely used program Modeller. Both programs use direction vectors from peptide backbone atoms and attach newly calculated amino acid side chains to the established template [18].

DockoMatic directs the manipulation of the ligand PDB file to prepare it for amino acid side chain replacement, through submission to TreePack, following a five step process (see Figure 2.5): A) the residue of interest and the two surrounding amino acids are copied into a new PDB file; B) the side chain atoms of the excised tripeptide are stripped from the analog PDB file, the backbone atoms and the beta carbon atom are retained; C) the amino acid at the point of mutation is replaced to create the peptide analog; D) the analog tripeptide file is submitted to TreePack, which uses the backbone atoms in concert with beta carbon atoms to form point vectors for the new side chains (except in the case of glycine, which does not have a beta carbon atom); E) the desired side chains are then extracted from the TreePack modified analog PDB file to be grafted back into the original ligand file, adjusting the remaining atoms to account for atom numbering differences.

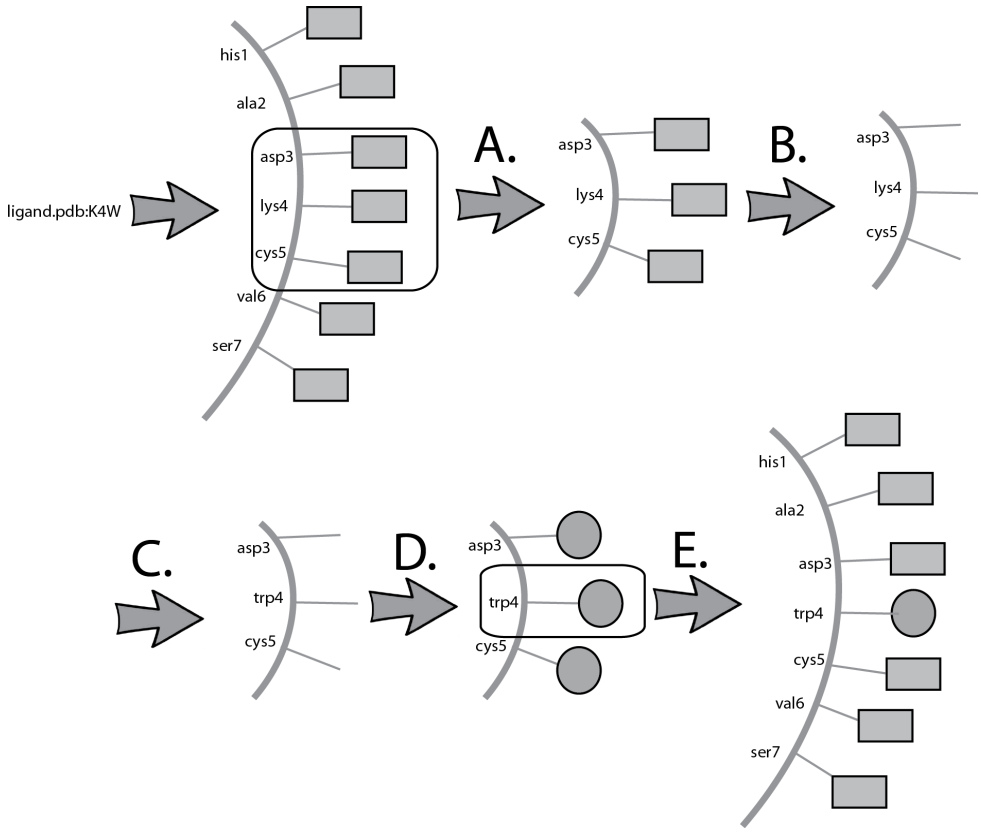

Figure 2.5 DockoMatic Peptide Analog Creation. TreePack peptide manipulation process for ligand site-directed amino acid substitution PDB file creation. From DockoMatic initiated command to produce ligand.pdb:K4W, the following five steps take place: A) the residue of interest (Lys4) and the two surrounding amino acids (Asp3 and Cys5) are copied into a new PDB file; B) the side chain atoms of the excised tripeptide are stripped from the analog PDB file, the backbone atoms and the beta carbon atom are retained; C) the amino acid at the point of mutation is replaced to create the peptide analog (Lys4→Trp4); D) the analog tripeptide file is submitted to TreePack, which uses the backbone atoms in concert with beta carbon atoms to form point vectors for the new side chains; E) the desired side chains are then extracted from the TreePack modified analog PDB file to be grafted back into the original ligand file.

This process may be repeated as many times as is necessary to form the desired mutated ligand, depending on whether a single or multiple point mutant is defined by the user. TreePack operates by first defining a bubble around the intended side chains to be modified. These bubbles are the parameters for the space available for the new side chain atoms. TreePack then minimizes the energy of the structure of the residue that is packed

in the bubble [54,55]. Three side chains are submitted to TreePack from the original ligand PDB file to minimize the potential for atomic overlap when the new residue is grafted back into the original ligand file.

High Throughput Virtual Screening (HTVS)

By itself AutoDock does not possess the functionality to efficiently setup and process the binding of multiple ligands to a receptor simultaneously, nor can it directly accommodate combinations of different ligands, receptors, and grid box locations. In order to ease setup of multiple jobs, DockoMatic processes lists of ligands and box coordinates for the desired receptor, followed by automatic job creation for each possible combination of ligand, receptor, and grid box coordinate. For example, supplying a list of 10 peptide ligands, one receptor, and three different box coordinate files results in (10 × 1 × 3 =) 30 different jobs being created. Through the DockoMatic interface, the user can then edit this job list, select jobs, and queue them for batch processing.

DockoMatic manages the submission of multiple ligand structures for binding to a receptor using swarm. Swarm allows DockoMatic to submit multiple jobs to a cluster simultaneously. The speed and efficiency of high throughput jobs is dependent upon the architectural constraints of the cluster. While DockoMatic can be run on a standalone workstation, it was designed to perform HTVS on a cluster, and as such, does not make assumptions regarding the number of jobs a user would wish to run on a single workstation at the same time, nor which job to run first. The most efficient use of DockoMatic on a single machine is to limit the maximum number of concurrent jobs started to the number of processors or cores in the computer.

An additional script has been written to ease HTVS studies where multiple peptide-analog ligand files need to be created. To generate a full complement of multiple point mutations manually would require thousands to millions of typed lines; each line would specify the desired mutation. This is difficult if not impossible to manage. A recursive algorithm was developed to automate the mutant list. The complete code for the algorithm has been included in Appendix A; the basic functionality of the algorithm is summarized below:

1) the ligand peptide sequence
2) path to the PDB structure file
3) comma separated numerical substitution positions (defaults to all)
4) cys flag (whether cys residues will be replaced)
5) polarity (whether it will be maintained, swapped, or random)
6) output file name (defaults to List.txt)

From the six components of the input, the HTVS algorithm processes each possible combination of point mutations that satisfy the user-specified parameters. The script utilizes three possible substitution sets, in which the first, polar set, contains the 11 amino acids that contain charge or polarity. The second is the non-polar set, with the remaining nine amino acids. Then there is the third set, the complete set, which contains all 20 amino acids. Which of these sets is used is dependent on the polarity setting as provided by the user. If the user elects to maintain polarity, then whichever set, polar or non-polar, the original amino acid is in is the set used for substitution. The same occurs for the polar substitution sites. If the user desires to swap the polarity, then the inverse is true, for an initial polar side chain, the new set of substitutions will be the non-polar, thus

swapping the substitution set used at that position. The final option provides the user with a random selection, or the amino acid substitution set is all possible combinations. For example, α-conotoxin (α-Ctx) MII with the sequence GCCSNPVCHLEHSNLC, represents the peptide we want modified. If we want to substitute six positions on the peptide, say N5, H9, L10, E11, H12, and L15, and maintain polarity, then the total number of peptides that would be created by the algorithm is 1,185,921. When a polar side chain is in the substitution position, there are 11 possible substitutions available. When it is non-polar, there are nine possible substitutions. So the total number of analogs would be the multiplication of the number of substitutions possible for each substitution site. So for this example, it would be 11 X 11 X 9 X 11 X 11 X 9, as sites 5, 9, 11, and 12 are polar with sites 10 and 15 being non-polar, creating the total from above.

Summary, Screening, and Analysis of Results by an Intuitive Process

DockoMatic parses, summarizes, and simplifies AutoDock results for the user. The results of AutoDock are output in the form of a single DLG file, with the size of the file dependent upon the number of simulations specified by the user. Summary output from DockoMatic includes separate ligand PDB files for each simulation in addition to a summary of the binding energy, inhibition constant, conformation statistics, and cluster rank. DockoMatic correlates the result information for each simulation into a single file that serves as the source file for data ranking. The PDB file with the highest rank (1 being the highest) represents the ligand to receptor combination with the lowest binding energy and is generally considered to be the most favorable binding model.

To further reduce the time required for data analyses, DockoMatic provides a results check button. This button was specifically designed for use with large grid

coordinates. For example, if the receptor binding site is unknown, a more general procedure of encompassing the entire receptor inside the search grid would allow for clusters of results. It is then possible to define a targeted GPF over the clusters, or suspected binding site, which can then be used by DockoMatic. From this second grid, DockoMatic screens the results and outputs the best and average values of both the estimated binding energy and the estimated inhibition constant. This is statistically useful to determine the location of potential binding sites. A greater number of results within the targeted GPF is an indication that the binding site has been properly identified. The output from this process includes: 1) the percent of runs where the ligand binds in the secondary GPF coordinates, 2) the average and best ΔG, and 3) the average and best K_i. This information is formatted in a simple text file similar to the ranked results list mentioned above.

For HTVS experiments, an analysis script (see Appendix B for full code) has been provided that will search through all results and generate a tab-separated file listing the result location, the ligand, and the lowest estimated ΔG. The input required for this script is simply the output path and the desired name of the list. It defaults to List.txt. Upon activation, the script will travel through each of the job directories searching for the top ranking result, extracting the relevant information, the ΔG for each ligand, and placing this information inside a hash table with the ligand as the key. The ligand serves as the key because there is likely no duplication in ligand names, whereas there may be duplication in the ΔG estimate. Once complete this hash table is sorted in rank order, and the results are written to the output file with the lowest ΔG listed first. This file is in a tab-separated format, so it can be opened in a multitude of spread sheet applications,

which saves the user time that it would take to manually extract the given information of the HTVS experiments.

Summary

This chapter has detailed the design and development of DockoMatic with six associated features: 1) an intuitive GUI, 2) AutoDock job setup, submission, and management of docking experiments, 3) creation of PDB files for linear peptide ligands, 4) peptide analog creation from a template PDB file, 5) HTVS, and 6) summary of results and analysis. AutoDock is a great tool for molecular docking studies; it consistently performs well and has been cited more than any other docking engine [43]. There are limitations to AutoDock that make it difficult to use for HTVS studies. Although there are many programs created that overcome this limitation (see Chapter 1), none of them contain all of the features we created on DockoMatic. DockoMatic was developed in collaboration between the Department of Computer Science and the Department of Chemistry and Biochemistry at Boise State University as a user friendly resource to enable undergraduate students the opportunity to perform HTVS studies. DockoMatic eliminates many of the mundane tasks required by AutoDock to perform molecular docking experiments. In our labs, DockoMatic has proven useful for all levels of users, from experienced to novice. All that is required from the user is the list of ligands, a receptor file, and a template grid box coordinate file. Once these have been submitted to DockoMatic, the push of a button will create peptide ligands, load required AutoDock files, select output directories, and begin processing of molecular docking calculations.

CHAPTER THREE: DOCKOMATIC – EXPERIMENTAL VERIFICATION AND VALIDATION

Introduction

Chapter One provided an overview of molecular docking and a list of open source and/or economical computational programs with GUI's for high throughput virtual screening (HTVS) docking experiments. In Chapter Two, the design and development of DockoMatic as an emerging resource for molecular docking was detailed. In the current chapter, I demonstrate how I validated the utility of DockoMatic for use in research application. DockoMatic is an intuitive GUI designed to facilitate job submission, and expand the capabilities of the widely used suite of automated docking tools collectively called AutoDock [3]. DockoMatic accepts PDB files of ligands and receptors with corresponding GPF files that specify the experimentally determined or predicted ligand binding domain on the receptor. A significant component of DockoMatic is the ability to enter a text file containing a list of peptide ligands for HTVS binding calculation in AutoDock. DockoMatic allows the user to enter peptide PDB files as ligands, and it can also create linear peptide ligand structure PDB files from strings of single letter amino acid code. The peptide ligands that are entered into DockoMatic are prepared for submission to AutoDock.

Each of DockoMatic's features has been experimentally validated. The linear ligand creation utility was tested with pentapeptide amino acid sequences. Running of

these experiments validated the ligand creation, submission, and management of AutoDock jobs by DockoMatic's ability to create, submit, and manage AutoDock jobs by DockoMatic. The analog creation feature was tested by the generation of conotoxin analogs. DockoMatic's HTVS capability was demonstrated with a mock experiment in which each of the file types (i.e. PDB, GPF, and DLG) were created and/or copied to each output destination directory, demonstrating successful HTVS file management by DockoMatic.

Feature Validation

Peptide-Based Linear Ligand Creation

The linear peptide ligand creation feature was developed for the purpose of discovering the biological activity of pentapeptides. We sought to screen thousands of peptides against a range of macromolecular receptors. A trial study consisting of five randomly selected pentapeptide ligands (CCMWF, CDCMW, CFWMW, CHMWW, and CHWWM) were created in DockoMatic. Two biomacromolecular receptors were chosen for this study, *Aplysia californica* acetylcholine binding protein (*Ac*-AChBP) and a homology model of α3β2 nicotinic acetylcholine receptor (nAChR). Out lab is interested in the ligand binding determinants to these receptors. Understanding how peptides bind to macromolecular receptors is expensive and time consuming by traditional molecular biology bench laboratory methods. Computer modeling has evolved as a useful way to study the interaction between peptide ligands and large biological receptors in a time-efficient and economical manner [56].

While tools exist to create peptide-ligand structure files, we are not aware of any that automate the process. Applications like Spartan, ChemDraw, and etc., require users to manually create the peptide ligand by placing and rotating individual amino acids using a mouse [52,57]. In order to create a peptide with the correct sequence, the user must first select each amino acid, from a group of the 20 common amino acids; second, the oxidation state of each peptide needs be set so that the amino terminus is an ammonium and the carboxy terminus is a carboxylate. The user then selects parameters for the program to create a three-dimensional coordinate structure for each peptide, saves the files, or later converts them into PDB file format.

A 61 node Beowulf cluster at Boise State University was used to test the ability of DockoMatic to automatically create linear peptide structure files. The files used for this test included: 1) the receptor PDB files derived from the crystal structure of *Ac*-AChBP, 2UZ6, and a homology model of α3β2 nAChR, and 2) five pentapeptide ligands with the following sequences: CCMWF, CDCMW, CFWMW, CHMWW, and CHWWM.

All five ligands were simultaneously submitted to DockoMatic in a single text file, with one peptide sequence in single letter amino acid code per line. DockoMatic successfully created the corresponding PDB structure files (see Figure 3.1). The DockoMatic GUI that accepts the ligand text file was then prompted to take the five ligand PDB files and automatically direct and pair them with the receptor PDB files and matching GPF files for submission to AutoDock for processing. Upon job completion, DockoMatic parsed the DLG files into individual result PDB files. These result files were ranked according to the estimated ΔG and were easily viewable by clicking on the "PyMol" button.

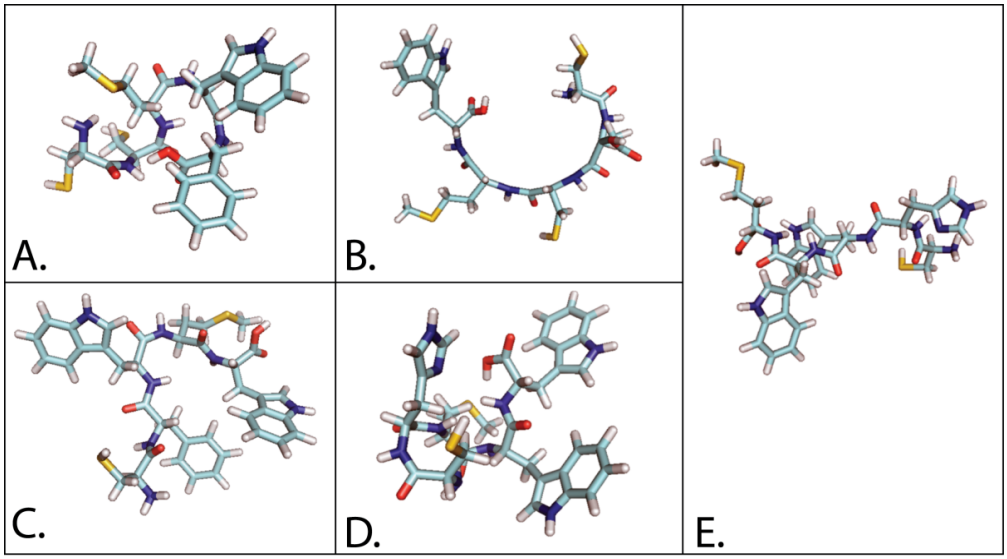

Figure 3.1 Linear pentapeptides created using DockoMatic by an automated process. A) CCMWF, B) CDCMW, C) CFWMW, D) CHMWW, and E) CHWWM.

Figure 3.2 shows an example of one of the created ligands, CCMWF, docked with the *Ac*-AChBP and α3β2 nAChR receptors as viewed by PyMol. The comparative binding energies between the five pentapeptides and the two receptors, as they appear in the results file, is displayed in Table 3.1.

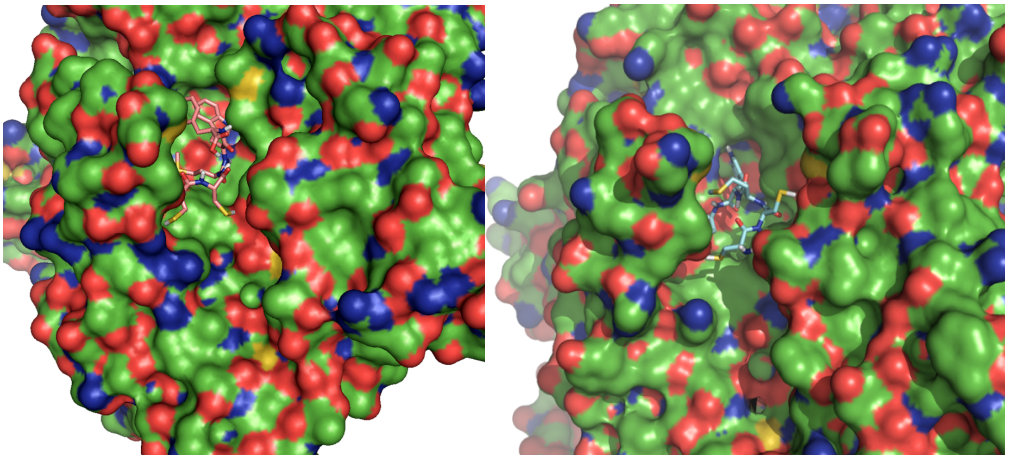

Figure 3.2 DockoMatic example output. DockoMatic result output PDB file image showing the best ranked (lowest ΔG) binding conformation for CCMWF in complex with *Ac*-AChBP (left) and α3β2 nAChR (right) as calculated by AutoDock, and visualized in PyMol.

Table 3.1 DockoMatic docking of select pentapeptides showing top energy results for *Ac*-AChBP and α3β2 nAChR as well as the difference in activity.

Pentapeptide	AChBP kcal/mol	α3β2 kcal/mol	Δ
CCMWF	-6.39	-6.84	0.45
CDCMW	-4.85	-6.41	1.56
CFWMW	-7.38	-10.59	3.21
CHMWW	-6.07	-9.69	3.62
CHWWM	-6.62	-7.92	1.3

Examination of the results shows that pentapeptide CFWMW and CHMWW have a remarkable difference in binding affinity for the two receptors, with CFWMW having -7.38 Kcal*mol^{-1} for *Ac*-AChBP and -10.59 Kcal*mol^{-1} for α3β2 nAChR. This shows a difference in binding affinity of 3.21 Kcal*mol^{-1}. In a similar manner the pentapeptide CHMWW has an estimated ΔG of -6.07 and -9.69 Kcal*mol^{-1} for *Ac*-AChBP and α3β2 nAChR, respectively; a difference in binding affinity of 3.62 Kcal*mol^{-1}. From this small trial of five pentapeptides, it is easy to see that DockoMatic can be used to accept a file containing a list of ligands, automate the structure file creation, initiate AutoDock calculations, and provide output that can be easily analyzed in table form or viewed using PyMol.

An experienced user can create the PDB file for a pentapeptide using Spartan, in approximately 2.5 minutes. It required approximately 12.5 minutes to create the five pentapeptide ligands for the trial just described. A user unfamiliar with molecular modeling software could take significantly longer to create these ligands. This time is dependent upon the length of the amino acid sequence, adding more amino acids generally causes the creation time to grow linearly with the number of amino acids.

In contrast, DockoMatic can prepare the same five pentapeptides in approximately 34 seconds of computational time. For DockoMatic, essentially no user time is required for ligand creation, regardless of the sequence length. All that DockoMatic requires is a single string representation of the amino acid sequence.

Create, Submit, and Manage AutoDock Jobs

Our tests showed that it took 31 minutes to perform all tasks required to submit the five pentapeptides for binding to the two receptors using AutoDock. This includes 12 minutes to create PDB files for the five pentapeptide ligands and 19 minutes to prepare the ligand, receptor, and GPF grid files as described in Chapter Two. In contrast, the time required to perform the same sequence of events using DockoMatic consists of the time to enter the location of the input files and press two buttons that create and start the jobs, which can be done in under one minute.

In this instance, with five amino acid strings listed in a ligand input file, one box coordinate file, and one receptor, it took DockoMatic approximately 16 seconds of user time to begin the five AutoDock jobs. Adding one minute to that time for grid box file creation yields a total time of 1 minute and 16 seconds.

Using AutoDock, a user must wait while the atom affinity map files are created. This process took approximately 19 minutes during our test. With DockoMatic, affinity map file creation is automated and requires a fraction of a minute of user time. Once the ligand, receptor, GPF grid, and the affinity map files are created and prepared, the time required to run a given AutoDock job is hardware dependent. So, comparative job runtimes are not particularly meaningful in the sense that they only show differences in hardware. More relevant than the time to run a few AutoDock jobs is the user time

required to manage and submit the AutoDock jobs, a complete breakdown of user time is shown in Table 3.2.

Table 3.2 Comparative user time between manual use of AutoDock and DockoMatic.

Task	Manual	DockoMatic
Create 5 pentapeptides	12 min	< 5 s
Prepare 5 Ligand Files	2.5 min	< 5 s
Prepare 5 Receptor Files	2.5 min	< 5 s
Create 5 GPFs	14 min	1 min
Total	31 min	1 min 16 s

Assuming the ligand list and the template box coordinate file have been generated, creating and running large numbers of jobs with DockoMatic takes essentially the same amount of user time as 1 job. The process involves browsing for the correct ligand, receptor, and grid box files followed by job submission. For instance, if using a list of 256 ligands, the only difference to the experiment above would be the name of the ligand list file.

Based on the previous experiment, attempting the same task of starting 256 docking jobs manually would require approximately 26 hours of user time before the jobs could be submitted to AutoDock. Ten of those hours would be dedicated to ligand creation alone, whereas with DockoMatic there is no additional user time required.

Peptide-Analog Structure Creation

The analog creation feature was validated using conotoxins as ligands. Conotoxins are small, 10-30 amino acid peptides that are cystine rich (i.e., contain multiple cysteine residues joined by disulfide bonds), and tend to be highly constrained

structurally. Conotoxins are among the most potent and selective ligands in their binding to myriad biological receptors, offering promise in the development of therapies for diseases including epilepsy, Parkinson's, Alzheimer's, schizophrenia, and many others [58,59]. Conotoxins can be broken down into superfamilies and further subdivided based on their Cysteine arrangement, cystine pattern, and target receptor [60]. For the purposes of evaluating the analog creation feature of DockoMatic, we chose to study the α-conotoxins (α-Ctxs) of the A-superfamily that selectively bind to nAChRs [61,62,63]. The cysteine-rich sequence and cystine composition of conotoxins results in structure rigidity a quality that is conducive to nuclear magnetic resonance (NMR) structure elucidation.

A comparative study was performed for α-Ctx structures obtained from the RCSB, elucidated by either NMR spectroscopy or X-ray crystallography versus the DockoMatic-generated peptide analog structures and their binding to three different *Ac*-AChBP structures. The specific ligands used for this study were: 1) ImI[R11E], PDB code 1E74; 2) ImI[R7L], PDB code 1E75; 3) ImI[D5N], PDB code 1E76; and 4) PnIA, PDB code 1PEN. The receptor models used for this study consisted of X-ray crystal structures: 1) *Ac*-AChBP with ImI bound, PDB code 2BYP; 2) *Ac*-AChBP with ImI bound, PDB code 2C9T; and 3) *Ac*-AChBP with PnIA[A10L:D14K], PDB code 2BR8. The *Ac*-AChBP PDB files were manually cleaned to remove bound ligand and water molecules. AChBPs are homopentameric proteins (see Figure 3.3). At the intersection of each subunit is a binding cavity. The result of this cleaning procedure is a structure consisting of only a pair of subunits containing the ligand binding domain where

conotoxins are found to be present in crystal structures of ligand/receptor complexes (see Figure 3.3). This structure was then used as the receptor for ligand binding studies.

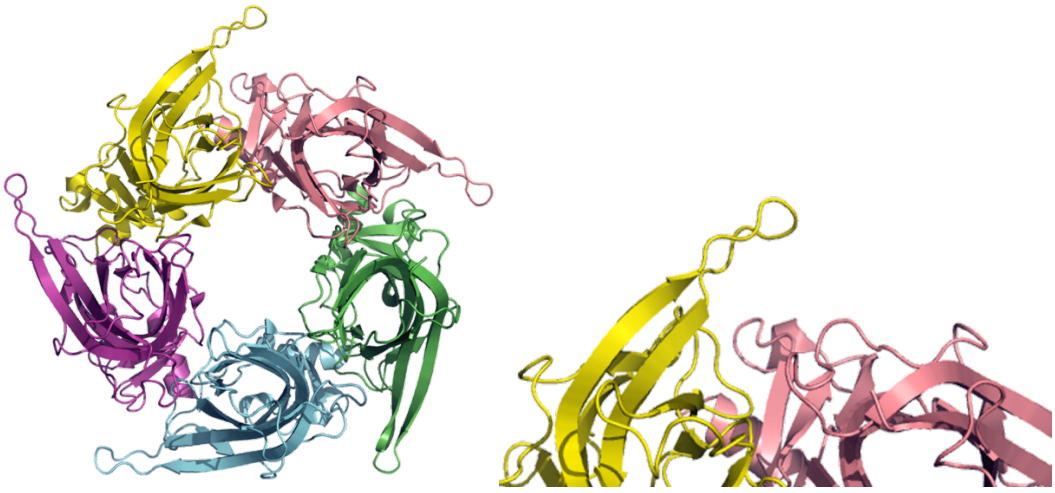

Figure 3.3 Structure of *Ac*-AChBP, showing all subunits (left) and cleaned subunit pair (right).

To access the analog utility in DockoMatic, the user enters the filename, including full location path, of the native RCSB retrieved peptide PDB file, followed by a colon, and then the desired substitution. To generate the α-Ctx ImI analog that substitutes an arginine amino acid in position eleven of the peptide with a glutamate, the user would enter: ImI.pdb:R11E. Each additional replacement is added and separated by colons (i.e., PnIA.pdb:A10L:D14K). DockoMatic further allows the user to submit list files of peptide analogs to AutoDock that begin by stating the original PDB file, with its location, followed by the substitutions separated by colons. A sample list using PnIA as the ligand and including various analogs is:

/home/username/conotoxins/PnIA.pdb

/home/username/conotoxins/PnIA.pdb:A10L

/home/username/conotoxins/PnIA.pdb:A10L:D14K

When submitted to DockoMatic, the above text list prepares three different ligands: one the original PDB file for α-Ctx PnIA, and the other two are analogs using the original structure as a template. This approach can be used to automate the submission of hundreds of analogs of a known ligand structure to accomplish HTVS of peptide ligands to a desired receptor.

The analysis of AutoDock binding results was performed to evaluate how DockoMatic/TreePack generated peptide analog structures bound to the *Ac*-AChBP receptor model compared to structures of the peptides independently determined by NMR spectroscopy or X-ray crystallography. By employing the result-check feature, the most energetically favorable conformations were compared and their backbone coordinates entered into the root mean square deviation (RMSD) calculator of the computer program, Visual Molecular Dynamics (VMD) [64]. A comparison of two structures with an RMSD under 2.0 Å indicates that bound ligand structures are in similar orientation.

Peptide analog structures, created through the DockoMatic GUI using the integrated TreePack software, were compared to structures of conotoxins deposited in the RCSB by two sets of experimental procedures. For experiment one, the first step required the creation of a model that provided an accurate depiction of a bound ligand to the *Ac*-AChBPs. To do this, crystal structures of *Ac*-AChBP receptors with bound ImI or the PnIA analog PnIA[A10L:D14K] were selected from the RCSB. The conotoxin ligand was computationally eliminated from the binding cavity of the *Ac*-AChBP followed by removal of water molecules from the receptor. Receptor cleaning eliminates the non-essential ligand and all water molecules. To test the ability of DockoMatic to run

jobs through AutoDock and generate reliable results, the conotoxin peptide ligand that was extracted from the ligand bound receptor/crystal structure complex was redocked into the now vacant (i.e., cleaned) receptor. Table 3.3 includes the results of ligand redocking with respect to ligand-binding orientation and binding energy as compared to the original crystalline structure.

Table 3.3 DockoMatic redocking of X-ray crystal and NMR solution structures with associated estimated binding energy and RMSD.

Receptor *Ac*-AChBP PDB codes	Ligand	Estimated Binding Energy kcal*mol^{-1}	RMSD Angstrom (Å)
2BR8	PnIA[A10L:D14K]	-15.44	1.01
2BYP	ImI	-16.03	0.88
2C9T	ImI	-13.87	1.22

For native ImI, there were two different crystal structures in the RCSB that were used for this exercise (i.e., 2BYP and 2C9T). Redocking of the extracted ImI peptide to the 2BYP receptor provided a peptide-ligand overlay between the crystal structure complex and the computationally determined binding complex with a rmsd of 0.88 Å. Conotoxin peptide redocking to the *Ac*-AChBP receptor model 2C9T yielded an overlay of bound peptide ligand between experimentally determined structure and computationally calculated structure of 1.22 Å. When the structure of the double mutant PnIA[A10L:D14K] was redocked into 2BR8, the RMSD was 1.01 Å. A visual representation of the bound ligand demonstrating the structural orientation of each peptide overlaid in the ligand binding domain of the receptor is shown in Figure 3.4. This result is significant because it demonstrates that the peptide sequence files entered into

DockoMatic, followed by submission to AutoDock, provide output that is within reasonable agreement to experimentally determined structure binding images (i.e., RMSD ≤ 1.2 Å).

One goal of this work is to scale the process by several orders of magnitude relative to the number of ligands that can be simultaneously submitted to AutoDock for binding calculations (i.e., high throughput screening of potential drug candidates). In this control experiment, we have validated the successful integration of AutoDock into DockoMatic to yield good homology between expected binding and computationally predicted binding to a common receptor [29].

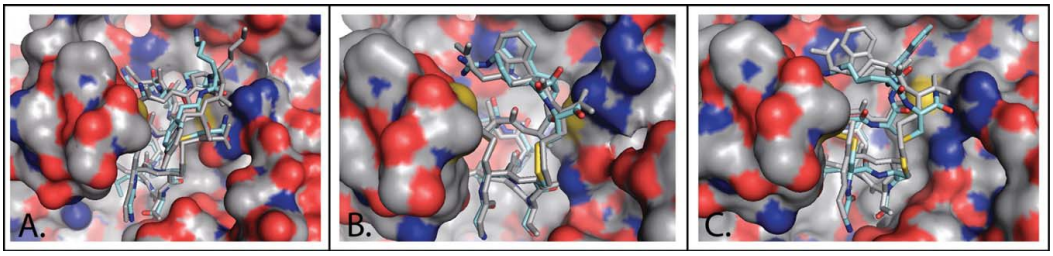

Figure 3.4 *Ac*-AChBP structures with ligand redocked using DockoMatic. Original ligand, grey, redocked ligand, blue. (A) PnIA[A10L:D14K] rebound to 2BR8 with an RMSD of 1.01 Å; (B) ImI redocked to 2BYP, RMSD of 0.88 Å; (C) ImI redocked to 2C9T, RMSD of 1.22 Å.

Next, validation of the TreePack driven utility for the automated creation of peptide analogs in DockoMatic was performed by comparing conotoxin analog structures generated by DockoMatic with NMR solution structures deposited in the RCSB. DockoMatic requires a parent peptide file for analog creation; the ligands submitted were the NMR solution structure PDB files for PnIA and ImI, PDB codes 1PEN and 1IMI, respectively. Three analogs of ImI: ImI[R11E], ImI[R7L], and ImI[D5N], and one analog

of PnIA: PnIA[A10L:D14K] were selected based on solution structure availability in the RCSB.

Two sets of experiments were conducted in parallel: 1) the solution structures were used as the ligands to bind to the *Ac*-AChBP receptor, and 2) the analog sequence was entered into DockoMatic, the ligand generated, and the ligand binding calculation automatically performed through AutoDock. The results were filtered using the result-check feature in DockoMatic, followed by orientation and docking conformation comparisons evaluated by calculated RMSD and estimated binding energy. Peptide ImI[R11E] was bound to three receptor crystal structures of *Ac*-AChBP, 2BR8, 2BYP, and 2C9T, with calculated binding energy of -12.16, -10.67, and -10.62 Kcal*mol^{-1}, respectively (Table 3.4). The overlay of the same analog generated in DockoMatic resulted in ligand binding energies of -11.13, -11.88, and -10.58 Kcal*mol^{-1} to the three *Ac*-AChBPs, showing a difference in energy of 1.03, 1.21, and 0.04, respectively. The RMSD of the three receptor overlays for peptide ImI[R11E] are 0.94, 0.56, and 0.81 Å, respectively.

The result of the redocking experiment with ImI analog, ImI[R7L] as compared to the DockoMatic generated docking of the TreePack created ImI[R7L] with the three *Ac*-AChBP receptors produced energy differences of 0.15, 0.17, and 0.89 Kcal*mol^{-1}, and demonstrated ligand RMSDs of 1.82, 0.76, and 0.85 Å. The same experiment using peptide ImI[D5N] provided energy differences of 0.67, 2.04, and 3.46 Kcal*mol^{-1} respectively, and RMSD differences of 1.18, 1.47, and 1.15 Å, respectively. For final comparison, the extracted ligand analog of PnIA, PnIA[A10L:D14K] was redocked to the *Ac*-AChBP 2BR8 with a calculated binding energy of -15.44 Kcal*mol^{-1}. The same

experiment was performed with the structure of native PnIA, 1PEN, yielding a binding energy of -14.6 Kcal*mol^{-1}, thus a difference of 0.84 and an RMSD between bound ligands of 0.43 Å.

Table 3.4 **Comparative results listing the estimated binding energies. Column 1 lists the receptor PDB codes for each experiment. Column 2 lists the α-Ctx ligands tested against column 1. Columns 3 and 4 list the results of the NMR solution structure and the DockoMatic created structure of the α-Ctx ligands. Columns 5 and 6 compare the results of the experiment by standard deviation of the estimated binding energy and the RMSD of the backbone structures.**

Receptor *Ac*-AChBP PDB code	α-Ctx Ligands*	NMR Solution Structure Estimated Binding Energy (kcal*mol^{-1})	DockoMatic^{+} Structure Estimated Binding Energy (kcal*mol^{-1})	Standard Deviation Comparing Estimated Binding Energy	RMSD (Å)
2BR8	PnIA[A10 L:D14K]	-15.44	-14.60	0.59	0.43
2BR8	ImI[R11E]	-12.16	-11.13	0.73	0.94
	ImI[R7L]	-11.09	-11.24	0.11	1.82
	ImI[D5N]	-13.80	-13.13	0.47	1.18
2BYP	ImI[R11E]	-10.67	-11.88	0.86	0.56
	ImI[R7L]	-12.59	-12.76	0.12	0.76
	ImI[D5N]	-14.88	-12.84	1.44	1.47
2C9T	ImI[R11E]	-10.62	-10.58	0.03	0.81
	ImI[R7L]	-13.49	-12.66	0.59	0.85
	ImI[D5N]	-15.54	-12.08	2.45	1.15
ImI[D5N] 1E76.					
$^{+}$PDB codes for DockoMatic templates: PnIA 1PEN, ImI 1Im1					

The RMSD of all conotoxin peptide ligand structures ranged from 0.56 to 1.82 Å in binding comparisons between experimentally determined peptide structure and DockoMatic-created ligand structure analogs. The difference in estimated binding energy from matching poses varied by less than 3.5 Kcal*mol^{-1}. These results demonstrate that the TreePack analog creation tool in DockoMatic provides ligand structures that bind

with similar orientation and affinity to RSCB structures determined experimentally. DockoMatic offers the ability to generate analog structures for peptides in a fraction of the time and expense of experimentally determined peptide ligand structures.

High Throughput Virtual Screening (HTVS)

The HTVS functionality of DockoMatic was tested by mock experiment; the file types (i.e., PDB, GPF, and DLG) were created and/or copied to corresponding output directories. The purpose of this experiment was to evaluate whether DockoMatic could be used to process hundreds to thousands of jobs simultaneously, and to verify that job-specific files were organized correctly without loss or corruption. We chose not to test the HTVS capability against a specific biological system for these studies because our goal was to validate the ability of the software to manage the large numbers of files required for and generated from HTVS. The time required to actually run the molecular docking calculations in a HTV screen is dependent on the site and complexity of the ligand/receptor structure files, the number of docking calculations, and the computational capabilities of the cluster. It was the purpose of this investigation to demonstrate the ability of DockoMatic for HTVS, not to evaluate the computer hardware it is run on.

There are a number of factors that must be considered to determine the high-throughput capacity and limitations of DockoMatic. For example, the maximum number of jobs that can be submitted as a set from DockoMatic is dependent on both the number of subdirectories a given file system can accommodate, and the amount of disk space that is available to store results. For most file systems, there is a maximum number of subdirectories that can be created within each directory; for instance, the most common file systems used with a Linux kernel, ext3 and ext4 are limited to 32,000 and 64,000

subdirectories, respectively. This file system limitation can become a factor because each AutoDock trial in a set of jobs uses one subdirectory for output, and DockoMatic typically creates the output subdirectories for a set of jobs in a single output directory. The device currently used by our laboratory for storing output results uses Lustre, which is a parallel file system designed for use in a clustered environment. Lustre allows a maximum of 25 million subdirectories, although in practice this is not the predominant limitation. For instance, disk space becomes an issue for large numbers of jobs. Each DockoMatic job generates output files that are an average of 115 MB in size. At 115 MB, 1 million jobs requires on the order of 115 Terabytes of disk space, which easily exceeds the current raw capacity of 72 Terabytes of the Lustre file system.

Each ligand specified as input by the user in DockoMatic is automatically submitted to the cluster for processing as an AutoDock job. The AutoDock job performs a default of 100 ligand to receptor binding calculations, though this number can be changed by the user, and compiles the output into a single DLG file. For each completed AutoDock job, DockoMatic extracts in priority order the 100 receptor binding calculations into the PDB reference file. DockoMatic determines that an AutoDock job is complete when the DLG and the PDB reference files are created. Thus, for 1 ligand, 100 results are summarized and listed in a single DLG file, and the results are put into a user-specified rank order in the PDB reference file (e.g., from lowest to highest binding energy). Thus, DockoMatic incurs a very small amount of computation time to setup a job and submit it for processing as well as a relatively small amount of computation (relative to the total job run-time) time to parse, process, and summarize each completed job. It is important to note that DockoMatic processes each job as it completes (i.e., it

does not wait for all submitted jobs to complete before beginning the process of summarizing results).

There are also other factors that can be equally as important, such as the speed of the machine that is used to host DockoMatic, the amount of system memory available, the computational capacity of the cluster itself, and etc. Although any assessment of capacity is dependent on the setup and environment, it is important to have an estimation of limitation to assess the feasibility of a potential set of experiments using DockoMatic.

The DockoMatic GUI performs two primary computational functions for each AutoDock job. These are the following:

1. DockoMatic directs the flow of jobs by creating a folder for the assigned AutoDock output and then starts the AutoDock job.
2. DockoMatic monitors the output directory of each AutoDock job for the presence of a reference file that contains a summary of the job statistics.

The DockoMatic GUI is unaware of the process used to create the output file it is looking for, so it is not necessary to initiate AutoDock jobs in order to evaluate throughput limitations of the DockoMatic GUI. To evaluate the high throughput capacity of the software, a series of mock jobs were created to populate the output folders with the required reference file, rather than running actual AutoDock experiments. The evaluation was performed in this manner due to time constraints; the goal was to test how many jobs DockoMatic could adequately handle, regardless of job type. All aspects and functionality of DockoMatic were preserved. The experiment worked by the following series of steps: (1) a list of jobs was submitted to DockoMatic, (2) DockoMatic created all output folders, (3) DockoMatic populated the monitoring grid, listing each job and the

output location, (4) each job (in this case a file copy) was submitted to swarm and queued to the cluster with its status changed to ‘‘Started,’’ and (5) after the file was copied, DockoMatic recognized each job as being complete and the status was changed to ‘‘Done.’’ This process was timed for jobs ranging from 100 to 1,000,000 submissions (Table 3.5). For mock job lists of 100 and 1000, it took DockoMatic less than 1 s to create directories and populate management grids. The process of distributing files and acknowledging job completion required 51 s and 8.7 min, respectively. To initiate, 10,000 jobs required on the order of 8 s with the final recognition of all jobs completed just over an hour and a half later (1.57 hours). The time to set up 100,000 jobs and submit them was on the order of 231 s with an estimated completion time on the order of 15 hours. A trial consisting of 1,000,000 jobs is estimated to take on the order of 1 week of computer cluster time. In summary, in its current configuration, DockoMatic can reasonably handle the submission of 10,000–100,000 jobs for binding calculation.

Table 3.5 Evaluation of the high throughput capability of DockoMatic. Mock experiment results showing the number of jobs in each trial, the length of time to submit jobs, and the job completion time based on output file preparation.

Number of Jobs	Initiation time (s)*	Completion time$
100	<1	51 s
1,000	<1	8.7 min
10,000	8	1.57 hours
100,000	231	~15 hours&

*Time required to create job submission directories and populate grid boxes. $Time required to copy DLG and PDB reference files into output directories. &Estimated time.

Summary

DockoMatic is a standalone utility consisting of an intuitive GUI that can be used for the following purposes: (1) create linear peptide ligands; (2) create, manage, and submit AutoDock jobs; (3) produce analogs based on structure templates; and (4) perform high throughput submission to AutoDock. It was demonstrated that DockoMatic is efficient in handling automatic linear peptide ligand creation, through creating and validating small pentapeptides for bioactivity investigations [29]. The user controlled pipeline necessary to use AutoDock has been greatly reduced and simplified into a single button click by DockoMatic, to assist in using molecular docking in current laboratory research [29]. The expanded functionality of DockoMatic to perform *in silico* site-directed mutagenesis using the TreePack utility offers the opportunity for chemists and biologists to apply the extraordinary tools developed by computer scientists toward predictive science [30]. AutoDock has been successfully integrated into DockoMatic for the routine submission of hundreds to thousands of jobs with more possible based on computer cluster access, and the limitation of file system architectures. DockoMatic is freeware that calls on other freeware software (TreePack, OpenBabel, MGLTools, PyMol, and AutoDock) for successful integration of command line applications into a simple and elegant GUI.

Future Direction

Experimental Investigations/Publications

- Proteomics: Work in progress investigating pentapeptides and their bioactivity in various systems by inverse virtual screening.

- DockoMatic in education: Creation of a molecular docking lab exercise for use in the undergraduate curriculum; students will analyze the published crystal structure of the α-Ctx TxIA bound to the Ac-AChBP, propose analogs that will bind better based on their analysis, create the analogs, perform the molecular docking calculation, and explain the results using an analysis of atomic interactions between their analog and the receptor. This thesis will also serve as a tutorial for new users to DockoMatic that will be uploaded to the DockoMatic Wiki.
- HTVS α3β2 nAChR: Using DockoMatic's peptide-analog generation capability to run thousands of α-Ctx MII analogs against a homology model of α3β2 nAChR to investigate binding determinants.
- Collagen XI α1: Using DockoMatic to manage molecular docking studies to investigate collagen XI α1 interactions to extracellular matrix proteins.
- Collagen XI α1/OSM: Using molecular docking, via DockoMatic, to investigate potential binding patterns between collagen XI and oncostatin M (OSM).
- DockoMatic 2.0: New release of DockoMatic built on a java netbeans platform. Responsible for testing and validating the homology modeling wizard, Timely Integrated Modeller (TIM) for the creation of the Collagen XI NPP domain.

Software Updates

- Multiple Operating Systems: Increase DockoMatic's usefulness in any lab environment, independent of operating system.

- Third Party Scoring: Include a plugin to direct rescoring of AutoDock results through a standalone software scoring program such as XScore or Medusa.
- Automatic Installation Wizard: Increase the ease of installation of DockoMatic by including a step-by-step wizard that will direct the installation of DockoMatic and automatic detection/installation of its dependencies.
- Cyclic Peptide Prediction: Create cyclic peptides based on an amino acid sequence and predict most likely cystine bridges based on minimum energy end result.
- Macromolecular System Calibration: Create a calibration wizard that can take a set of known ligands with experimentally determined binding values and calibrate the computational results to narrow the gap between experimental and predictive results.
- Receptor Point Mutation: Apply the same technique, i.e. TreePack, used in peptide-analog generation to receptors.
- AutoDock Vina: Include the ability to use AutoDock Vina and other docking engines.
- Bond Distance analysis: Measure the distances of ligand atoms to those of the receptors to identify and catalog inter- and intramolecular interactions.

REFERENCES

1. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, et al. (2000) The Protein Data Bank. Nucleic Acids Res 28: 235-242.

2. Kuntz ID, Blaney JM, Oatley SJ, Langridge R, Ferrin TE (1982) A geometric approach to macromolecule-ligand interactions. J Mol Biol 161: 269-288.

3. Morris GM, Goodsell DS, Halliday RS, Huey R, Hart WE, et al. (1998) Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. J Comput Chem 19: 1639.

4. Goodsell DS, Olson AJ (1990) Automated docking of substrates to proteins by simulated annealing. Proteins 8: 195-202.

5. Wang R, Lai L, Wang S (2002) Further development and validation of empirical scoring functions for structure-based binding affinity prediction. J Comput-Aided Mol Des 16: 11-26.

6. Yin S, Biedermannova L, Vondrasek J, Dokholyan NV (2008) MedusaScore: An Accurate Force Field-Based Scoring Function for Virtual Drug Screening. J Chem Inf Model 48: 1656-1662.

7. Jiang X, Kumar K, Hu X, Wallqvist A, Reifman J (2008) DOVIS 2.0: an efficient and easy to use parallel virtual screening tool based on AutoDock 4.0. Chem Cent J 2.

8. Zhang S, Kumar K, Jiang X, Wallqvist A, Reifman J (2008) DOVIS: an implementation for high-throughput virtual screening using AutoDock. BMC bioinformatics 9.

9. Prakhov ND, Chernorudskiy AL, Gainullin MR (2010) VSDocker: a tool for parallel high-throughput virtual screening using AutoDock on Windows-based computer clusters. Bioinformatics 26: 1374-1375.

10. Trott O, Olson AJ (2010) Software news and update AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. J Comput Chem 31: 455-461.

11. Sanner MF (1999) Python: a programming language for software integration and development. J Mol Graph Model 17: 57-61.

12. Chushak Y, Stone MO (2009) In silico selection of RNA aptamers. Nucleic Acids Res 37: e87.

13. Jiang X, Kumar K, Hu X, Wallqvist A, Reifman J (2008) DOVIS 2.0.

14. Irwin JJ, Shoichet BK (2004) ZINC − A Free Database of Commercially Available Compounds for Virtual Screening. J Chem Inf Model 45: 177-182.

15. Collignon B, Schulz R, Smith JC, Baudry J (2011) Task-parallel message passing interface implementation of Autodock4 for docking of very large databases of compounds using high-performance super-computers. J Comput Chem 32: 1202-1209.

16. Thompson JD, Higgins DG, Gibson TJ (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Ress 22: 4673-4680.

17. O'Sullivan O, Suhre K, Abergel C, Higgins DG, Notredame C (2004) 3DCoffee: combining protein sequences and structures within multiple sequence alignments. J Mol Biol 340: 385-395.

18. Sali A, Blundell T (1993) Comparative Protein Modelling by Satisfaction of Spatial Restraints. J Mol Biol 234: 779-815.

19. Muegge I, Martin YC (1999) ARTICLES - A General and Fast Scoring Function for Protein -- Ligand Interactions: A Simplified Potential Approach. J Med Chem 42: 791.

20. Wang R, Liu L, Lai L, Tang Y (1998) SCORE: A New Empirical Method for Estimating the Binding Affinity of a Protein-Ligand Complex. J Mol Model 4: 379-394.

21. Eldridge MD, Murray CW, Auton TR, Paolini GV, Mee RP (1997) Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. J Comput-Aided Mol Des 11: 425-445.

22. Böhm HJ (1994) The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. J Comput-Aided Mol Des 8: 243-256.

23. Zhang C, Liu S, Zhu Q, Zhou Y (2005) A Knowledge-Based Energy Function for Protein−Ligand, Protein−Protein, and Protein−DNA Complexes. J Med Chem 48: 2325-2335.

24. Hu JZ, Bai L, Chen DG, Xu QT, Southerland WM (2010) Computational investigation of the anti-HIV activity of Chinese medicinal formula Three-Huang Powder. Interdiscip Sci 2: 151-156.

25. Goldgur Y, Craigie R, Cohen GH, Fujiwara T, Yoshinaga T, et al. (1999) Structure of the HIV-1 integrase catalytic domain complexed with an inhibitor: A platform for antiviral drug design. Proc Natl Acad Sci USA 96: 13040-13043.

26. Hu Z, Southerland W (2007) WinDock: structure-based drug discovery on Windows-based PCs. J Comput Chem 28: 2347-2351.

27. Vaqué M, Arola A, Aliagas C, Pujadas G (2006) BDT: an easy-to-use front-end application for automation of massive docking tasks and complex docking strategies with AutoDock. Bioinformatics 22: 1803-1804.

28. Streiff JH, Jones KA (2008) Volatile Anesthetic Binding to Proteins Is Influenced by Solvent and Aliphatic Residues. J Chem Inf Model 48: 2066-2073.

29. Bullock CW, Jacob RB, McDougal OM, Hampikian G, Andersen T (2010) Dockomatic - automated ligand creation and docking. BMC Research Notes 3.

30. Jacob RB, Bullock CW, Andersen T, McDougal OM (2011) DockoMatic: Automated peptide analog creation for high throughput virtual screening. J Comput Chem 32: 2936-2941.

31. Albuquerque EX, Alkondon M, Pereira EFR, Castro NG, Schrattenholz A, et al. (1997) Properties of Neuronal Nicotinic Acetylcholine Receptors: Pharmacological Characterization and Modulation of Synaptic Function. J Pharmacol Exp Ther 280: 1117-1136.

32. Wolf LK (2009) digital briefs: New software and Websites for the Chemical Enterprise. C&EN 87.

33. Ren J, Williams N, Clementi L, Krishnan S, Li WW (2010) Opal web services for biomedical applications. Nucleic Acids Res 38: W724-W731.

34. Johnson Graham T, Autin L, Goodsell David S, Sanner Michel F, Olson Arthur J (2011) ePMV Embeds Molecular Modeling into Professional Animation Software Environments. Structure (London, England : 1993) 19: 293-303.

35. Suvannang N, Nantasenamat C, Isarankura-Na-Ayudhya C, Prachayasittikul V (2011) Molecular Docking of Aromatase Inhibitors. Molecules 16: 3597-3617.

36. Fontham ETH, Thun MJ, Ward E, Balch AJ, Delancey JOL, et al. (2009) American Cancer Society Perspectives on Environmental Factors and Cancer. CA Cancer J Clin 59: 343-351.

37. Virtua Drug Ltd (2009) DockingServer. Available: http://www.dockingserver.com. Accessed December 2010.

38. Bikadi Z, Hazai E (2009) Application of the PM6 semi-empirical method to modeling proteins enhances docking accuracy of AutoDock. J Cheminf 1: 15.

39. Cai X, Bikadi Z, Ni Z, Lee E-W, Wang H, et al. (2010) Role of Basic Residues within or near the Predicted Transmembrane Helix 2 of the Human Breast Cancer Resistance Protein in Drug Transport. J Pharmacol Exp Ther 333: 670-681.

40. Abreu R, Froufe H, Queiroz M, Ferreira I (2010) MOLA: a bootable, self-configuring system for virtual screening using AutoDock4/Vina on computer clusters. J Cheminf 2: 10.

41. Shrodinger L (2009) The PyMOL Molecular Grphics System. Version 1.3 ed. Available: http://www.pymol.org. Accessed March 2009.

42. Li C, Xu L, Wolan D, Wilson L, Olson A (2004) AutoDock Tools: Virtual Screening of Human 5-Aminoimidazole-4-carboxamide Ribonucleotide Transformylase against the NCI Diversity Set by Use of AutoDock to Identify Novel Nonfolate Inhibitors. J Med Chem 47: 6881 - 6690.

43. Kitchen DB, Decornez H, Furr JR, Bajorath J (2004) Docking and scoring in virtual screening for drug discovery: methods and applications. Nature reviews Drug discovery 3: 935-949.

44. Kontoyianni M, McClellan L, Sokol G (2004) Evaluation of docking performance: comparative data on docking algorithms. J Med Chem 47: 558 - 565.

45. Chemical Computing Group (2010) MOE. Available: http://www.chemchomp.com.

46. Gold (2010) Version 1.2. Available: http://www.ccdc.cam.acuk/products/life_sciences/gold/.

47. Ewing TJA, Makino S, Skillman AG, Kuntz ID (2001) DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases. J Comput-Aided Mol Des 15: 411-428.

48. Friesner RA, Banks JL, Murphy RB, Halgren TA, Klicic JJ, et al. (2004) Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. J Med Chem 47: 1739.

49. Cavasotto CN (2006) Ligand Docking and Virtual Screening in Structure-Based Drug Discovery. AIP conference proceedings 851: 34-49.

50. Pearlman DA, Case DA, Caldwell JW, Ross WS, Cheatham TE, et al. (1995) AMBER, a package of computer programs for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to simulate the structural and energetic properties of molecules. Comput Phys Commun 91: 1-41.

51. Li C, Xu L, Wolan D, Wilson I, Olson A (2004) Virtual screening of human 5-aminoimidazole-4-carboxamide ribonucleotide transformylase against the NCI diversity set by use of AutoDock to identify novel nonfolate inhibitors. J Med Chem 47: 6681 - 6690.

52. Wavefunction, Inc. (2003) SPARTAN. Version 2004. Availabe: http://www.wavefun.com/products/spartan.html

53. The Open Babel Package. (2011) Version 2.2.3. Available: http://www.openbabel.org. Accessed: December 2011.

54. Xu J (2005) Rapid Protein Side-Chain Packing via Tree Decomposition. In: Miyano S, Mesirov J, Kasif S, Istrail S, Pevzner P et al., editors. RECOMB: Springer Berlin / Heidelberg. pp. 423-439.

55. Xu J, Berger B (2006) Fast and accurate algorithms for protein side-chain packing. J ACM 53: 533-557.

56. Zoete V, Grosdidier A, Michielin O (2009) Docking, virtual high throughput screening and in silico fragment-based drug design. J Cell Mol Med 13: 238 - 248.

57. Mills N (2006) ChemDraw Ultra 10.0 CambridgeSoft, 100 CambridgePark Drive, Cambridge, MA 02140. www.cambridgesoft.com. Commercial Price: $1910 for download, $2150 for CD-ROM; Academic Price: $710 for download, $800 for CD-ROM. JACS 128: 13649-13650.

58. Dutertre S, Lewis RJ (2006) Toxin insights into nicotinic acetylcholine receptors. BPharm 72: 661-670.

59. Millard EL, Daly NL, Craik DJ (2004) MINIREVIEW: Structure-activity relationships of α-conotoxins targeting neuronal nicotinic acetylcholine receptors. EJB 271: 2320-2326.

60. Jacob RB, McDougal OM (2010) The M-superfamily of conotoxins: a review. CMLS: 17.

61. Dutertre S, Lewis RJ (2004) MINIREVIEW: Computational approaches to understand α-conotoxin interactions at neuronal nicotinic receptors. EJB 271: 2327-2334.

62. Groebe DR, Dumm JM, Levitan ES, Abramson SN (1995) alpha-Conotoxins selectively inhibit one of the two acetylcholine binding sites of nicotinic receptors. Mol Pharmacol 48: 105-111.

63. McIntosh JM, Azam L, Staheli S, Dowell C, Lindstrom JM, et al. (2004) Analogs of a-Conotoxin MII Are Selective for a6-Containing Nicotinic Acetylcholine Receptors. Mol Pharmacol 65: 944.

64. Humphrey W, Dalke A, Schulten K (1996) VMD: Visual molecular dynamics. J Mol Graph 14: 33-38.

APPENDIX A

Recursive Algorithm for Analog Generation

```
#!/usr/bin/python
#Filename: analogs.py
import getopt, sys

#Polarity Sets
nonpolar = set(["A","G","I","L","M","F","W","V"])
polar = set(["R","N","D","E","Q","H","K","S","T","Y", "C"])
all = set (["A","G","I","L","M","F","W","V", "R","N","D","E","Q","H","K","S","T","Y",
"C"])

#Default variable values
sequence = "CCCCNCCCHLEHCCLC" #Owens adjusted Sequence
#sequence = "GCCSNPVCHLEHSNLC" #Actual Sequence
positions = []
polarity = ‘’
path = "/home/rjacob/MIIScreen/MII.pdb"
cflag = 1
filename = "List.txt"

#Main program function
def main ():
        try:
                opt, arg = getopt.getopt(sys.argv[1:], "s:n:f:cp:ho:")
        except getopt.GetoptError, err:
                usage()
                sys.exit(2)

        for o, a in opt:
                if o in ('-s'):
                        sequence = a
                        positions = range(len(sequence)+1)
                        positions.pop([0])
#                       print positions
                elif o in ('-n'):
                        positions = a.split(',')
                elif o in ('-c'):
                        cflag = 0
```

```
                elif o in ('-p'):
                        polarity = a
                elif o in ('-h'):
                        usage()
                elif o in ('-o'):
                        filename = a
                elif o in ('-f'):
                        path = a
                else
                        usage()

        fout = open(filename, "w")
        analog(sequence,path,positions,"",fout, polarity, cflag)
        fout.close()
        fout = open(filename, "r")
        splitFile(fout)


def usage():
        print "USAGE: analogs.py -s <sequence> [-n <substitution positions> -p
<Polarity> -o <output file> -c -h]\n"
        print "\t-s single letter amino acid sequence\n"
        print "\t-n comma seperated numerical substitution positions\n"
        print "\t-p Polarity (I)nverted, (M)aintained or (R)andom\n"
        print "\t-o output filename\n"
        print "\t-c turn on substitution of cys residues\n"
        print "\t-h displays this help message\n"


def analog (sequence, path, positions, adjust, file, polarity, cflag=1):
        num = positions.pop([0])-1
        letter = sequence[num]

        if (cflag==1 and (cmp(letter,"C")==0)):
                if (num < (len(sequence)-1)):
                        analog(sequence,path,positions,adjust,file,polarity, cflag)
                else:
                        file.write(path+adjust+"\n")
        elif (((letter in nonpolar) and (cmp(polarity,"M")==0)) or ((letter in polar) and
(cmp(polarity,"I")==0))):
                for i in nonpolar:
                        if (num < (len(sequence)-1)):
                                test=":" + letter + str(num+1) + i
                                if (cmp(letter,i)==0):

        analog(sequence,path,positions,adjust,file,polarity,cflag)
```

```
                                else:

        analog(sequence,path,positions,adjust+test,file,polarity,cflag)
                        else :
                                test=":" + letter+ str(num+1)+i
                                if (cmp(letter,i)==0):
                                        file.write(path+adjust+"\n")
                                else:
                                        file.write(path+adjust+test+"\n")
        elif (((letter in nonpolar) and (cmp(polarity,"I")==0)) or ((letter in polar) and
(cmp(polarity,"M")==0))):
                for j in polar:
                        if (num < (len(sequence)-1)):
                                test=":" + letter + str(num+1) + j

                                if (cmp(letter,j)==0):

        analog(sequence,path,positions,adjust,file,polarity,cflag)
                                else:
                                        analog(sequence,
path,positions,adjust+test,file,polarity,cflag)
                        else:
                                test= ":" + letter + str(num+1) + j
                                if (cmp(letter,j)==0):
                                        file.write(path+adjust+"\n")
                                else:
                                        file.write(path+adjust+test+"\n")
        elif (cmp(polarity,"R")==0):
                for j in all:
                        if (num < (len(sequence)-1)):
                                test=":" + letter + str(num+1) + j

                                if (cmp(letter,j)==0):

        analog(sequence,path,positions,adjust,file,polarity,cflag)
                                else:
                                        analog(sequence,
path,positions,adjust+test,file,polarity,cflag)
                        else:
                                test= ":" + letter + str(num+1) + j
                                if (cmp(letter,j)==0):
                                        file.write(path+adjust+"\n")
                                else:
                                        file.write(path+adjust+test+"\n")
        else:
                file.write("Not a valid amino acid\n")
```

```
def splitFile (file):
        numLines = 0
        nfiles = 1
        while 1:
                lines = file.readlines(100000)
                if not lines:
                        break
                files = "MIIanalogs%02d.txt"%nfiles
                newFile = open(files,"w")
                for line in lines:
                        newFile.write(line)
                newFile.close()
                nfiles = nfiles+1

if __name__ == "__main__":
        main()
```

APPENDIX B

Analysis Algorithm for HTVS

```
#!/usr/bin/perl
#This script will analyze the docking results for HTVS experiments.
#use String::Util qw(trim);



$mainDir = $ARGV[0];
@temp = split(//,$mainDir);
if ($temp[-1] eq "/") {chop($mainDir);}
print $mainDir."\n";
$myFile = $ARGV[1];

my($analog)="";
my %hash = ();
my $aDir;



chdir($mainDir) or die "Can't enter specified directory: $!\n";
opendir(DIR, ".") or die "Can't open the specified directory: $!\n";
@names = sort readdir(DIR) or die "Unable to read current directory:$!\n";
closedir(DIR);

foreach $name (@names) {
        next if (!($name =~ m/dock_\d+/)); #Skip everything that's not a dock folder
        opendir(ANA,$name) or die "This didn't work:$name\t$!\n";
        @files = readdir(ANA);
        $analog="";
```

```
	foreach $file (@files){
		if ($file =~ m/rank_1\.pdb/) {
			#print "$file\n";
			$analog = $file;
			break;
		}
	}
	$analog =$name  ."/".$analog;
	print $analog."\n";
	open(FILE,$analog) or $num=5000;
	while (<FILE>) {
		if ($_=~ /Binding\s+\=(.+)Kcal/) {
			$num = $1;
			$num =~ s/\s+//g;
			break;}
	}
	$aName = $mainDir."/".$analog;
	$hash{$aName}=$num;

	close(FILE);
	closedir(ANA);
}

open (MYFILE, ">>$myFile");
$hash{$a} <=> $hash{$b};
foreach $key (sort hashSort( keys(%hash))){
	print MYFILE "$key\t\t$hash{$key}\n";
}

sub hashSort {
	$hash{$a} <=> $hash{$b};
}
```