2022-12-01

What Makes GPCRs From Different Families Bind To The Same Ligand?

Kwabena Owusu Dankwah
University of Texas at El Paso

WHAT MAKES GPCRS FROM DIFFERENT FAMILIES BIND TO THE SAME LIGAND?

KWABENA OWUSU DANKWAH

Doctoral Program in Computational Science

APPROVED:

Ming-Ying Leung, Ph.D., Chair

Jonathon E. Mohl, Ph.D.

Charlotte Vines, Ph.D.

Shrikanth Gadad, Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School

Dedication

To my father

Yaw Karikari Dankwa

and my mother

Mary Darkwa

WHAT MAKES GPCRS FROM DIFFERENT FAMILIES BIND TO THE SAME LIGAND?

by

KWABENA OWUSU DANKWAH, M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

December 2022

Acknowledgements

I would like to first thank God for His grace upon my life. Secondly, I would like to express my gratitude to my advisor, Dr. Ming-Ying Leung of the Computational Science Department at The University of Texas at El Paso, for her advice, encouragement, enduring patience and constant support. Her guidance, belief in me and constantly driving me with energy and giving me her time in spite of anything else that was going on has helped me to give my best effort and be successful. I also wish to thank the other members of my committee Dr. Jonathon E. Mohl, Dr. Charlotte Vines, and Dr. Shrikanth Gadad for their encouragement, insightful comments, and questions. My heartfelt gratitude also goes to the research group members; Dr. Khodeza Begum and Dr. Fredrick Ayivor for their enormous participation in putting together this work. I would like to thank my father Yaw Karikari Daankwa, mother, Mary Darkwa and my siblings Anti Dankwa, Agyabeng Dankwa, Nketia Dankwa, Lydia Afrah and my aunty Madam Afua Biama Dankwa for their prayers and support. I thank my pastors LP. Marie Josette Williams and Ps. Derrick Williams for their prayers and support. Also, I am grateful for the support from the Computational Science Program and the NIH Grant #5G12MD007592 to the UTEP Border Biomedical Research Center.

Abstract

G protein-coupled receptors (GPCRs) are the largest class of cell-surface receptor proteins with important functions in signal transduction and often serve as therapeutic drug targets. With the rapidly growing public data on three-dimensional (3D) structures of GPCRs and GPCR-ligand interactions, computational prediction of GPCR ligand binding becomes a practical option for high throughput screening and other experimental approaches during the beginning phases of ligand discovery. In this work, we set out to computationally uncover and understand the binding of a single ligand to GPCRs from several different families. We analyzed the sequences and 3D structures of GPCRs from various families that bind to the same ligand. To conduct the analysis, we used currently available tools as well as newly developed Python scripts. These include MEME for motif search, FATCAT for 3D structural comparison, P2Rank for pocket prediction, APoc for pocket comparison, and our own Python codes for computing overlap scores. Comparison of 3D GPCR structures that bind to the same ligand revealed local 3D structural similarities and the similar regions often overlap with locations of binding pockets. Using Apoc, these pockets were found to be similar based on backbone geometry and side-chain orientation, and they correlate positively with electrostatic properties of the pockets. Moreover, the more similar the pockets, the more likely a ligand binding to the pockets will interact with similar residues, have similar conformations, and produce similar binding affinities across the pockets. These findings can lead to improved protein function inference, drug repurposing, and drug toxicity prediction, which can, in turn, accelerate the development of new therapeutics. Furthermore, the computational workflow and program codes established for this analysis can be developed into a software pipeline for more extensive investigation of GPCR-ligand binding mechanisms.

Table of Contents

Dedication iii

Acknowledgements v

Abstract vi

Table of Contents vii

List of Tables xi

List of Figures xii

Chapter 1: Introduction 1

1.1 GPCR structural characteristics 2

1.2 Classification of GPCRs 3

1.3 Ligands binding to GPCRs 4

1.4 Research objectives 6

Chapter 2: Literature Review 9

2.1 GPCR and Ligand Databases 9

2.1.1 UniProtKB/Swiss-Prot 9

2.1.2 RCSB PDB 10

2.1.3 BindingDB 10

2.1.4 IUPHAR/BPS 10

2.1.5 GLASS 11

2.1.6 GPCR-PEnDB 11

2.2 Protein Sequence Motif search 12

2.3 Protein-Ligand Binding and Docking 12

2.3.1 Binding pocket prediction 14

2.3.1.1 Template-based method 15

2.3.1.2 Geometry-based method 16

2.3.1.3 Energy-based method 16

2.3.1.4 Propensity-based method 17

2.3.1.5 Combination-based and others 17

2.3.2 Binding pocket comparison 18

2.3.3 GPCR ligand binding prediction 19
2.3.4 Protein-ligand docking 20
2.4 Binding pocket electrostatic properties 22
2.5 Protein Structural comparison 23
Chapter 3: Materials and Methods 25
3.1 Dataset collection 25
3.1.1 Data Sources 26
3.1.2 Procedures for data acquisition 27
3.1.3 Exploratory data analysis and data restructuring 28
3.2 Motif search 29
3.3 Structural comparison 30
3.4 Binding pocket prediction 31
3.5 Binding pocket comparison 31
3.6 GPCR ligand docking 32
3.7 Ligand binding pose and conformation 33
3.8 Protein ligand interaction 34
3.9 Predicted pocket and 3D structural similarity comparison overlap 34
3.10 Pockets electrostatic properties 35
Chapter 4: Results and Discussion 37
4.1 Currently Known 3D Structures of GPCR 37
4.2 GPCR ligand binding 38
4.3 Conserved motifs 42
4.4 Structural comparison 44
4.5 Binding pocket comparison and GPCR ligand docking relationship 46
4.6 Ligand binding pose and conformation 48
4.7 Protein ligand interaction and pocket electrostatic properties 50
4.8 Predicted pocket and 3D structural similarity comparison overlap 52
Chapter 5: Conclusion and Future Work 54
5.1 Conclusion 54
5.2 Future research and proposed approaches 54

References 56

Appendices 73

Appendix A: Tables 73

Appendix B: Procedure for workflow 76

Appendix C: Data Collection and Analysis 86

C.1: BindingDB Data 86

C.2: GLASS Data 86

C.3: IUPHAR Data 87

C.4: Combining BindingDB, GLASS, and IUPHAR Data Sets 87

C.5: Analysis of GPCR Ligand Interaction Data 93

C.6: Ligand 3D Structures 95

C.6.1: Convert from mol2 to pdbqt 95

C.6.2: Convert from sdf to pdbqt 95

C.6.3: Convert Ligand SMILES into 3D structures 95

C.7: Frequency distribution of GPCRs 3D structures 95

Appendix D: Motif Search 97

D.1: Cut Sequence into regions 97

D.2: Cut Sequence into Extracellular loops 97

D.3: Cut Sequence into Modified Extracellular loops 99

D.4: Run MEME 100

Appendix E: Analysis on 3D structures of GPCRs 100

E.1: 3D Structural Comparisons 100

E.1.1: Running FATCAT 100

E.1.2: Extract Superimposed 3D Part 101

E.2: Binding Pocket Prediction and Comparisons 107

E.2.1: Binding Pocket Prediction 107

E.2.2: Binding Pocket Comparison 110

E.3: Pockets and 3D Superimposed Part Overlap Scores 113

Appendix F: GPCR Ligand Docking and Ligand Pose and Conformation 116

F.1: Run AutoDock Vina 116

F.2: Parse and Gather Binding Affinities from Docking 117

F.3: Docked Ligand Conformation 118

Appendix G: Analysis of results 119

Curriculum Vita 128

List of Tables

Table 1: Number of GPCR subfamilies, sub-subfamilies, subtypes, and sequences in the extended IUPHAR and GRAFS classification families [Begum et al., 2020]. 4
Table 2: Frequency distribution of ligands by the number of distinct IUPHAR families they bind to. 39
Table 3: List of the 11 ligands that bind to 3 different IUPHAR families 39
Table 4: Ligands that binds to GPCRs of three IUPHAR families 40
Table 5: List of Ligands and the Proteins they bind to for the Control Dataset 41
Table 6: Human conserved motifs of GPCR sequences across three IUPHAR families found by the MEME system 43
Table 7: Pairwise structural comparison of human GPCR using FATCAT 46
Table 8: Summary statistics for PS-scores for all possible pairs of pockets for each ligand. 47
Table 9. Correlation between PS-score and absolute difference of the binding affinities 48
Table 10: Correlation between PS-scores and Chebyshev distances of MS-WHIM scores for all compared pocket pairs by ligand. 52
Table 11: Human conserved motifs of GPCR sequences across two IUPHAR families found by the MEME system 73

List of Figures

Figure 1: G protein-coupled receptor without ligand [C. Vines, Personal Communication, November 7, 2020] 3
Figure 2: Crystal Structure of mGluR5 with ligands bound (green) to it, PDB ID: 6FFI [Sehnal et al., 2018; Christopher et al., 2018] 6
Figure 3: Docking of a small molecule (green) into the crystal structure of the beta-2 adrenergic G-protein coupled receptor (PDB: 3SN6, source: Wikipedia: https://en.wikipedia.org/wiki/Docking_(molecular)) 14
Figure 4: Schematic illustration of docking a small molecule ligand (green) to a protein target (black) producing a stable complex (source: Wikipedia: https://en.wikipedia.org/wiki/Docking_(molecular)) 21
Figure 5: FATCAT Structural alignment of PDB: 3PBL and PDB: 6CM4 (Left: Flexible, right: Rigid) 24
Figure 6: Workflow overview 25
Figure 7: Frequency distribution of GPCRs 3D structures in PDB classified by family 37
Figure 8: Frequency distribution of new GPCR 3D structures added by year 38
Figure 9: A plot of the $RMSD_{Actual}$ and $RMSD_{Docked}$ pairs of the pockets. 49
Figure 10: Ligand binding poses and conformations across the different GPCRs binding pockets. 50
Figure 11: (**A**) AJLF GPCR interaction in the pocket. (**B**) Pocket electrostatic properties with AJLF docked into the pocket. 51
Figure 12: 3D structure of 3G04 vs 7LCK in-bound with the ligand USZP. The ligand USZP is docked to a predicted binding pocket. Red part of the GPCRs is where they are 3D structurally similar. 53

Chapter 1: Introduction

G protein-coupled receptors (GPCRs) are the largest class of cell-surface receptors (membrane proteins) [Singh et al., 2019] and are encoded by more than 800 genes in the human genome [Zhang et al., 2015]. Over 50% of drugs approved by the United States Food and Drug Administration (FDA) target integral membrane proteins [Goodsell et al., 2019]. Most of these drug targets belong to four well-studied protein families (GPCRs: 30%; voltage-gated ion channels: 8%; ligand-gated ion channels: 7%; and transporters: 7%) [Goodsell et al., 2019]. GPCRs make up the bulk as they are involved in a various of physiological processes including vision, taste, smell, inflammation, cell recognition, pheromone signaling and more.

GPCRs play an essential role in intracellular signaling. They are of clinical importance in many diseases, including cancer [Jo & Jung, 2016], which have crucial implications for tumor growth and metastasis [Lappano & Maggiolini, 2012]. Various molecules like hormones, lipids, peptides and neurotransmitters exert their biological effects by binding to GPCRs coupled to heterotrimeric G-proteins, which are highly specialized transducers capable of modulating various signaling pathways [Lappano & Maggiolini, 2012].

Insel et al., (2018) used TaqMan qPCR arrays to quantify the mRNA expression of ~340 GPCRs and found that human chronic lymphocytic leukemia (CLL) cells, breast cancer cell lines, colon cancer cell lines, pancreatic ductal adenocarcinoma (PDAC) cells, cancer-associated fibroblasts (CAFs), and PDAC tumors express 50 to over 100 GPCRs, including many orphan GPCRs that lack known physiologic agonists. These authors proposed that highly expressed GPCRs in cancer cells (for example, GPRC5A in PDAC and colon cancer cells and GPR68 in PDAC CAFs) may contribute to the malignant phenotype and could serve as biomarkers or potential novel therapeutic targets for cancer treatment.

Besides cancer, GPCRs are also involved in other diseases and their treatment. For example, GPCRs serve as targets for anabolic drugs in osteoporosis [Diepenhorst et al., 2018]. GPCRs have been implicated in the pathogenesis of Alzheimer's disease (AD) and in multiple stages of the hydrolytic processing of the amyloid protein precursor (APP), a precursor protein involved in the formation of amyloid plaques in the brain of patients with AD [Zhao et al., 2016]. Accumulated data have shown that GPCRs can bind to β-secretase (β-site APP cleaving enzyme 1, BACE1) and γ-secretase, critical enzymes in the hydrolytic processing of APP [Zhao et al., 2016]. Similarly, GPCRs are implicated in the pathophysiology of diverse neurodegenerative diseases which include frontotemporal dementia, vascular dementia, Parkinson's disease, and Huntington's disease [Huang et al., 2017].

Freudenberg et al. (2018) suggest that galanin, an endogenous ligand for the GPCR galanin receptor type 2 (GALR2), plays an essential role in epilepsy, confirming an earlier notion that galanin is a potential target in the treatment of epilepsy [Mazarati et al., 2001]. In particular, galanin depletion from the hippocampus may contribute to the maintenance of seizure activity [Clynen et al., 2014], and there may be genetic evidence showing that a galanin loss-of-function mutation leads to epilepsy in humans [Guipponi et al., 2014]. This is one of the examples indicating that understanding GPCRs ligand binding could have significant impacts on the modern medical field.

1.1 GPCR STRUCTURAL CHARACTERISTICS

All GPCRs have the same characteristic molecular structure consisting of an N-terminus, a cytoplasmic C-terminus along and seven hydrophobic transmembranes (7TM) domains connected by three intracellular and three extracellular loops (Figure 1).

Figure 1: G protein-coupled receptor without ligand [C. Vines, Personal Communication, November 7, 2020]

Interesting insights into the impact of the three-dimensional (3D) structures of essential target proteins for new drug discovery (e.g. GPCR) have been revealed through open access to structural data [Goodsell et al., 2019]. These include GPCR ligand binding modes, G-protein binding mechanisms, structural similarity and diversity of GPCR ligand recognition, GPCR functional states, and properties of a receptor structure competent for G-protein binding [Zhang et al., 2015].

1.2 Classification of GPCRs

Several classification systems have been used to classify the superfamily of GPCR proteins. One of the most widely used systems is that of the International Union of Basic and Clinical Pharmacology (IUPHAR) [Horn et al., 2003], which divided GPCRs into six major classes, A (rhodopsin-like), B (secretin receptor family), C (metabotropic glutamate), D (fungal mating pheromone receptors), E (cyclic AMP receptors), and F (frizzled/smoothened) based on sequence homology and functional similarity.

Another system of classification is the GRAFS system. The GRAFS system clusters GPCRs in five main families: Glutamate, Rhodopsin, Adhesion, Frizzled/Taste2, and Secretin [Schiöth & Fredriksson, 2005]. The main difference between the IUPHAR system and the GRAFS system is that IUPHAR is designed to cover all GPCRs, in both vertebrates and invertebrates,

whereas GRAFS is designed for mammalian species. Table 1 shows of the number of GPCRs in each class of GPCRs stored in the GPCR-PEnDB [Begum et al., 2020].

GPCR-PEn (GPCR Prediction Ensemble) is a web server developed by our team which utilizes sequence similarity, transmembrane structure, and dipeptide composition to determine if a protein sequence is a GPCR (www.gpcr.utep.edu). It has a database component called GPCR-PEnDB (GPCR Prediction Ensemble Database), and a searchable MySQL database of confirmed GPCRs and non-GPCRs [Begum et al., 2020]. Our team constructed it to allow users to access helpful information on GPCRs in various organisms conveniently and to compile reliable training and testing datasets for different combinations of computational tools [Begum et al., 2020].

Table 1: Number of GPCR subfamilies, sub-subfamilies, subtypes, and sequences in the extended IUPHAR and GRAFS classification families [Begum et al., 2020].

IUPHAR	GRAFS	Subfamilies	Sub-subfamilies	Subtypes	No. of Sequences
Class A	Rhodopsin-like	11	61	287	2493
Class B	Adhesion-like	1	5	15	91
	Secretin-like	1	9	33	113
Class C	Glutamate-like	4	5	22	112
Class D	Fungal pheromone*	1	1	1	13
Class E	cAMP receptor*	1	1	1	11
Class F	Frizzled	1	1	11	82
Class T2R**	Taste2 receptor**	1	1	25	211

*Not in the original GRAFS system

**Not in the original IUPHAR or GRAFS system

1.3 LIGANDS BINDING TO GPCRS

GPCRs bind to G proteins inside and outside the cell to ligands such as ions, biogenic amines, peptides, hormones, growth factors, lipids and photons [Huang et al., 2017]. Figure 2 is

an example of a GPCR (Crystal Structure of mGluR5) 3D structure showing ligands (green) bound to it; however, there are some GPCRs for which no information about the ligands that bind to them is known yet [Gad & Balenga, 2020].

In this study, we obtained data on GPCR ligand interactions from three different public databases: BindingDB, GLASS, and IUPHAR. IUPHAR and BindingDB contain a collection of experimental protein-small molecule interactions and measured binding affinities, focusing chiefly on the interactions of proteins considered to be candidate drug-targets with ligands that are small, drug-like molecules [Gilson et al., 2016; Armstrong et al., 2020]. GLASS (GPCR-Ligand ASSociation) is considered to be the most comprehensive and up-to-date ligand association repository in the field and encompasses a broad range of GPCR-related pharmacological data, gathered from many data sources and PubMed literature mining [Chan et al., 2015].

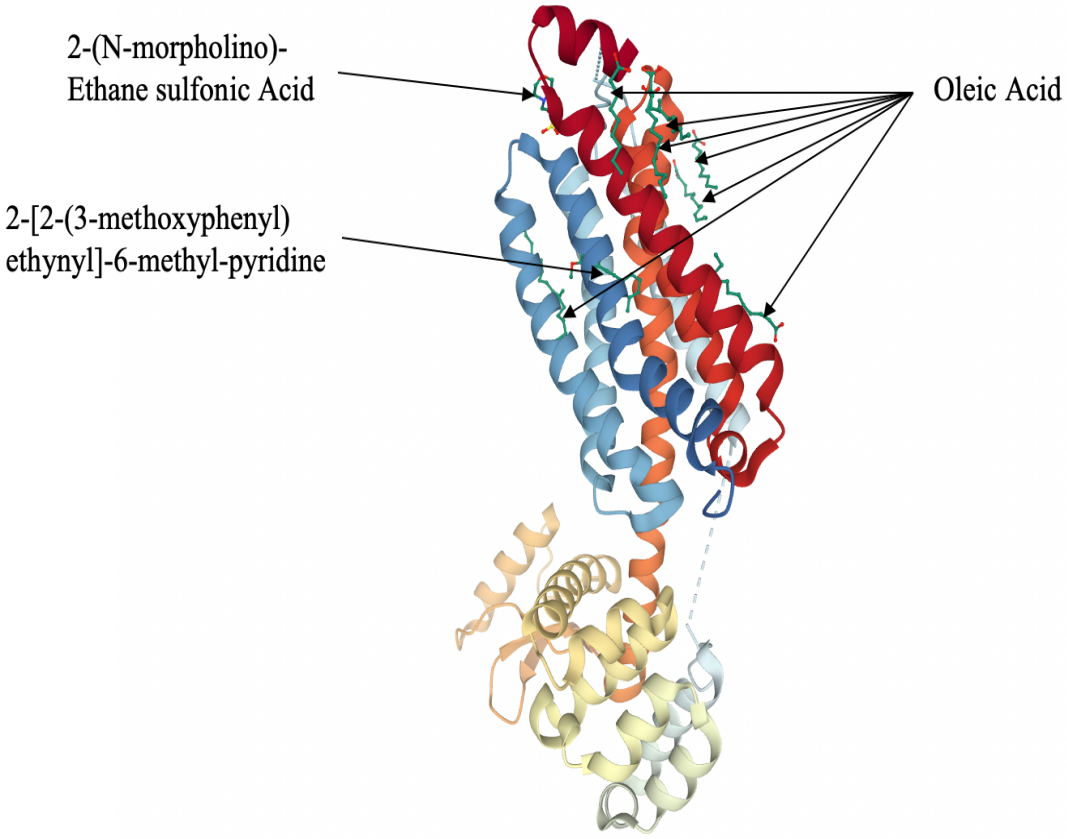


Figure 2: Crystal Structure of mGluR5 with ligands bound (green) to it, PDB ID: 6FFI [Sehnal et al., 2018; Christopher et al., 2018]

1.4 RESEARCH OBJECTIVES

The work aims to gain more insights into the sequence and structural features of GPCRs from different families that enable them to bind to the same ligand. Understanding how GPCRs bind to their ligands can help identify features that influence GPCR-ligand interactions, which in turn aids in fine-tuning of computational tools in selecting feature sets to predict GPCR-ligand binding. It can also provide information to help decide which computational tools to use. For

example, Seo et al. (2018) proposed a novel protein-ligand binding prediction method, which uses the local and global structure of a ligand and amino acid motif sequence of a GPCR. Their way infers hidden properties of good ligand-receptor binding encoded as a random forest classifier. Ciancetta et al. (2015) have shown that molecular dynamics can account for critical aspects such as a realistic microenvironment for GPCRs, the flexibility of GPCRs, and water molecule-mediated interactions that could play an essential role in the binding of ligands that are ignored by molecular docking. The inherent flexibility of GPCRs allows them to function through molecular interactions, changing their structural conformations in response to the presence of other molecules or changes in the environment [Teilum et al., 2009].

Recently, a number of machine learning algorithms have been developed to predict GPCR ligand binding, e.g., random forest, convolutional neural network, deep neural network, decision tree and support vector machine [Seo et al., 2018; Li et al., 2019; Raschka & Kaufman, 2020]. However, but they have been tested only on specific groups of GPCRs and none has focused on features that characterize the binding of a single ligand to multiple GPCR families.

Given all the information described above, our goal is to learn more about the sequence and structural features of GPCRs from different families that allow them to bind the same ligand. In particular, we wanted to help answer these questions:

1. Do the GPCRs that bind to the same ligand share any conserved sequence motifs? Are they locally similar in terms of their 3D structures?
2. For GPCRs that bind to the same ligand, how similar are their binding pockets in sequence and structure? Which residues of the GPCR interact with which atoms of the ligand?
3. Do ligands binding to human GPCRs from different families bind with the similar poses and affinities?

In Chapter 2 of this dissertation, I will present a review of current GPCR-ligand binding prediction methods. This is followed by a description of my work in answering the questions asked above and exploring computational tools in Chapter 3, results in Chapter 4, and conclusions on my

research findings in Chapter 5. The materials in this dissertation have been published as a conference paper [Dankwah et al., 2021] and a journal paper [Dankwah et al., 2022].

Chapter 2: Literature Review

To investigate the posted research questions in Chapter 1, we have developed a computational workflow that encompasses several steps involving protein sequence motif search, protein structure comparison, protein-ligand binding, and docking, the details of which will be described later. In this chapter, we first review the available public databases from which we gathered the GPCR and ligand data and the existing computational approaches and algorithms to be applied in several specific steps of our analyses.

2.1 GPCR AND LIGAND DATABASES

In this subsection, we provide information about the source of the data set we collected throughout the study.

2.1.1 UniProtKB/Swiss-Prot

The Universal Protein Resource (UniProt) is a comprehensive resource for protein sequence and annotation data [Bateman, 2019]. The UniProt Knowledgebase (UniProtKB) is the database of UniProt. This is the central hub for collecting useful protein information with accurate, consistent, and rich annotation [Bateman, 2019]. Over 95% of UniProtKB entries are derived from the coding sequences (CDS) translation, and submitted to the public nucleic acid databases, the EMBL-Bank/GenBank/DDBJ databases [Bateman, 2019]. Each entry contains mainly the amino acid sequence, protein name or description, taxonomic data and citation information [Bateman, 2019]. As of September 30, 2022, this database has 226,771,949 sequence entries available through TrEMBL. We obtained data on the start and end of the regional sequence (the N-terminal, extracellular loops, intracellular loops, the seven helices, and the C-terminal) of the GPCRs.

2.1.2 RCSB PDB

Protein Data Bank (PDB) is a single worldwide repository of information about the 3D structures of large biological molecules, including proteins and nucleic acids, which are found in all organisms including bacteria, yeast, plants, flies, other animals, and humans [Begum et al., 2020]. The structures in the PDB archive range from tiny proteins and bits of DNA to complex molecular machines like the ribosome [Begum et al., 2020]. The archive is freely available to users and is updated weekly. Under the leadership of Helen M. Berman, the Research Collaboratory for Structural Bioinformatics (RCSB) became responsible for the management of the PDB in 1998. As of September 30, 2022, PDB contains 195,858 structures.

2.1.3 BindingDB

BindingDB is a publicly accessible experimental protein-small molecule interaction database of measured binding affinities, focusing chiefly on the interactions of proteins considered to be candidate drug targets with small, drug-like molecules ligands [Gilson et al., 2016]. BindingDB data entries are primarily derived from scientific articles and, increasingly, US patents [Gilson et al., 2016]. These data come from various measurement techniques, including enzyme inhibition and kinetics, isothermal titration calorimetry, NMR, radioligand, and competition assays [Gilson et al., 2016]. As of September 30, 2022, BindingDB contains 2.6M data for 1.1M Compounds and 8.9K Targets; of those, 1,154K data for 533K Compounds and 4.4K Targets were manually curated by BindingDB curators.

2.1.4 IUPHAR/BPS

The International Union of Basic and Clinical Pharmacology (IUPHAR) and the British Pharmacological Society (BPS) have jointly developed the Guide to PHARMACOLOGY which is an open-access, expert-curated database of molecular interactions between ligands and their targets [Armstrong et al., 2020]. It is intended as a “one-stop shop” portal for pharmacological

information. Its main aim is to provide a searchable database with quantitative data on drug targets and the prescription medicines and experimental drugs that act on them [Armstrong et al., 2020]. G protein-coupled receptors are one of the six primary pharmacological targets into which the Guide is divided, with the others being: ion channels, nuclear hormone receptors, catalytic receptors, enzymes and transporters [Alexander et al., 2019]. This database contains 3,002 human targets, 1,611 of which with curated quantitative ligand interactions, 11,348 ligands, 8,396 of which have curated quantitative target interactions, 1,756 approved drugs, 1,052 with curated quantitative interactions, and many more, as of September 30, 2022.

2.1.5 GLASS

GLASS (GPCR-Ligand ASSociation) encompasses a wide breadth of GPCR-related pharmacological data, gathered from many data sources and PubMed literature mining [Chan et al., 2015]. This database contains 562,871 unique GPCR-ligand entries, 1,046,026 experimental data entries, 3,056 GPCR entries (of which 825 are human GPCR), and 342,539 ligand entries, as of September 30, 2022.

2.1.6 GPCR-PEnDB

GPCR-PEnDB (GPCR Prediction Ensemble Database) is a searchable MySQL database of confirmed GPCRs and non-GPCRs [Begum et al., 2020]. GPCR-PEnDB currently contains 3129 confirmed GPCR and 3575 non-GPCR sequences collected from the UniProtKB/Swiss-Prot protein database, encompassing over 1200 species [Begum et al., 2020]. The non-GPCR entries include transmembrane proteins for evaluating various prediction programs' abilities to distinguish GPCRs from other transmembrane proteins [Begum et al., 2020]. Each protein is linked to information about its source organism, classification, sequence lengths and composition, and other derived sequence features [Begum et al., 2020].

2.2 Protein Sequence Motif search

GPCRs retain various functional domains within and between species to bind various ligands, activate G-proteins, and participate in signaling pathways [Nagarathnam et al., 2011]. Most members of the rhodopsin-like GPCR family (class A) have several highly conserved motifs within their transmembrane domain (TMD), such as a DRY motif in TMD3 [Römpler et al., 2006]. Mutations in the DRY motif were found to increase the basal activity of mammalian orthologs of the chemoattractant receptor GPR33 in mouse and Gerbillinae species before the receptor was pseudogenized the Gerbillinae subfamily [Römpler et al., 2006]. The motif search algorithm searches for a set of similar subsequences in a group much longer sequences [Bailey et al., 2006]. One of the most commonly used tools to search for novel motifs in sets of biological sequences is MEME (Multiple EM for Motif Elicitation) [Bailey et al., 2006; Bailey et al., 2009; Bailey et al., 2015]. MEME works by searching for repeated, un-gapped sequence patterns that occur in the DNA or protein sequences.

2.3 Protein-Ligand Binding and Docking

Proteins play essential roles in all cellular activities including: enzyme catalysis, maintenance of cellular defenses, metabolism and catabolism, signaling within and between cells, and maintaining the structural integrity of cells [Roche & McGuffin, 2016; Daniel B. Roche et al., 2011; Roche et al., 2012; Roche et al., 2013]. The binding of ligands to proteins is essential for many vital processes in living organisms [Nguyen et al., 2017]. They are necessary for many proteins to function correctly [Kukol, 2014]. Interactions between proteins and ligands are necessary for signal transduction, immune response, and gene regulation [Fu et al., 2018]. Protein-ligand interaction studies are important to understand the mechanisms of biological regulation and provide a theoretical basis for the design and discovery of new drug targets [Fu et al., 2018].

GPCRs play a very important role that involves binding to ligands and activating signal transduction pathways and cellular responses [Seo et al., 2018]. For example, galanin, an endogenous ligand for the GPCR galanin receptor type 2 (GALR2), plays an important role in epilepsy [Freudenberg et al., 2018; Mazarati et al., 2001]. In humans, the initial phase of human visual perception includes photon retention by four distinctive visual pigments [Srinivasan et al., 2019]. These visual pigments include an apoprotein, opsin, that is covalently bound to the chromophore 11-cis-retinal (11CR), a vitamin A derivative that acts as an inverse agonist, locking the photoreceptor protein opsin into its inactive state [Srinivasan et al., 2019]. Similarly, inhaled selective β2-agonists (e.g., salbutamol, formoterol, indacaterol, etc.) are widely used in the treatment of obstructive airway diseases such as asthma [Matera et al., 2018]. These drugs bind to the β2-adrenoceptor (β2-AR) and cause the activation of certain G-proteins and subsequent generation of cyclic adenosine monophosphate (cAMP) in airway smooth muscle, resulting in bronchodilation [Matera et al., 2018]. Figure 3 shows a ligand (green) bound to a protein.

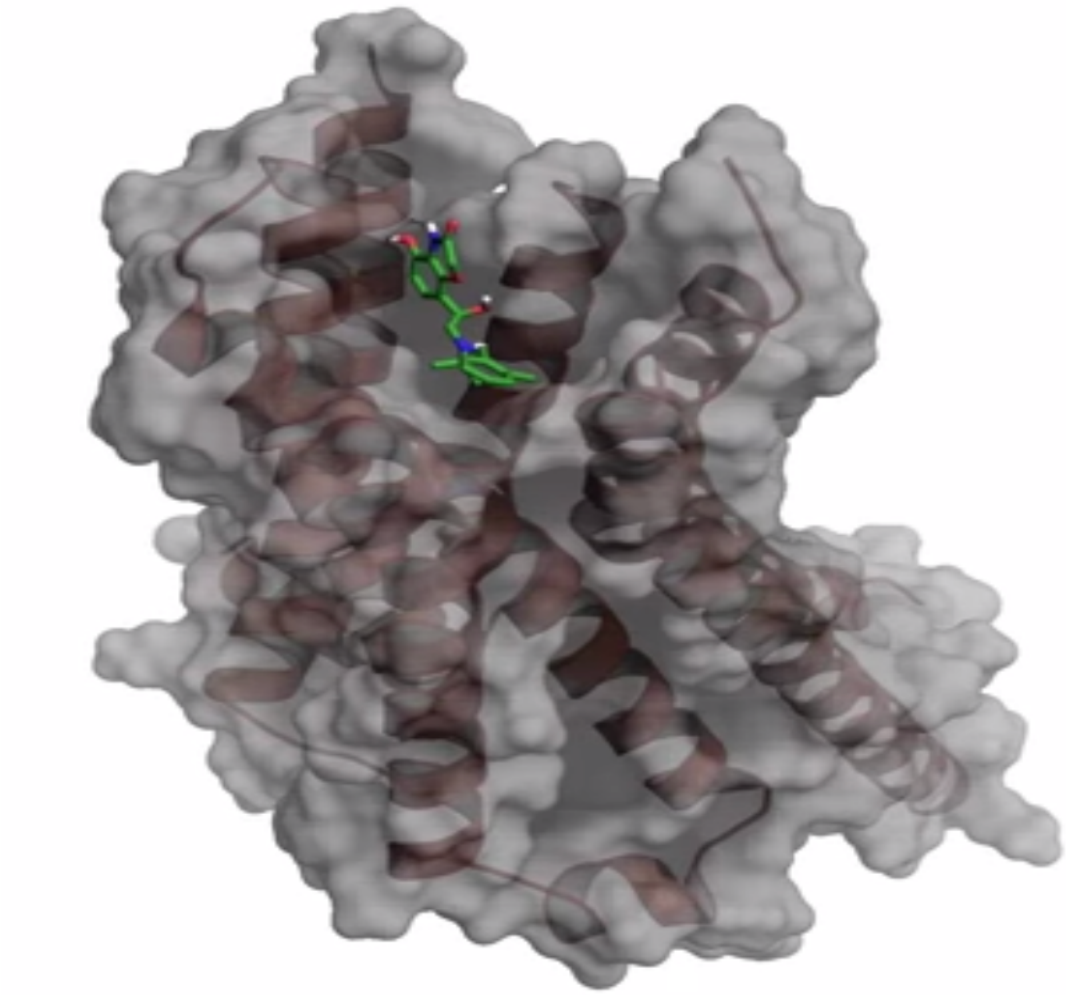


Figure 3: Docking of a small molecule (green) into the crystal structure of the beta-2 adrenergic G-protein coupled receptor (PDB: 3SN6, source: Wikipedia: https://en.wikipedia.org/wiki/Docking_(molecular))

2.3.1 Binding pocket prediction

The protein-ligand interaction is essential for many proteins to perform their biological function [Kukol, 2014]. This interaction is generally specified with respect to the ligand involved and the site at which the interaction occurs [Kukol, 2014]. Detection of ligand binding sites is often the starting point for protein function identification and drug discovery [Brylinski & Skolnick, 2008]. In order to gain insight into the interactions and thus into protein function and its influence on protein activity, efforts have been made to develop methods that can computationally predict

the ligand binding sites of proteins [Kukol, 2014]. Due to the site specificity of the ligand binding sites, these methods use one or more of the following types of properties to distinguish the binding site from other parts of the protein surface: evolutionary, geometric, energetic, and statistical [Kukol, 2014]. These methods have been categorized into the following: template-based, geometry-based, energy-based, propensity-based, combination-based, and others [Kukol, 2014].

2.3.1.1 Template-based method

Template-based methods use homologous and/or similar structures with known binding sites [Kukol, 2014]. Protein sequences are homologous if they descend, usually with divergence from a common ancestral sequence [Lee et al., 2007]. Homologues can be divided into orthologues and paralogues [Lee et al., 2007]. Orthologues are found in different species and have been separated by a speciation event rather than by gene duplication [Lee et al., 2007]. Paralogues are the product of gene duplication within a species, but since gene duplication can occur before speciation, paralogues can also exist across species [Lee et al., 2007].

The basic idea behind all template-based methods for predicting ligand binding sites is that proteins that share sequence homology are known to adopt similar 3D structures and typically perform similar biological functions [Lee et al., 2007]. First, we identify ligand-bound complex structures (to serve as a template for finding potential ligand-binding sites) that have sufficient sequence similarity to the target protein (the protein in which binding sites are to be predicted). Second, we overlay the target protein and identified templates, including information on known ligand-binding sites. Finally, a consensus ligand binding site can be revealed. Its characteristics as a putative ligand binding site for the target protein is evaluated by comparison to those of known ligand binding sites [Kukol, 2014]. There are several template-based methods. These include 3DLigandSite [Wass et al., 2010;Wass & Sternberg, 2009], FINDSITE [Brylinski & Skolnick, 2008; Skolnick & Brylinski, 2009], Firestar [López et al., 2007], I-TASSER [Zhang, 2007], ProBiS [Konc & Janežič, 2010], and IntFOLD [Daniel B. Roche et al., 2011].

These methods are similar in principle (i.e., they all use templates) but differ in implementation. For example, 3DLigandSite combined the use of the predicted structure of the targets with both residue conservation and localization of ligands bound to homologous structures [Wass et al., 2010;Wass & Sternberg, 2009] whereas FINDSITE is a method predicting ligand binding sites and functional annotation based on the similarity of binding sites between sets of weakly homologous template structures identified from threading [Brylinski & Skolnick, 2008; Skolnick & Brylinski, 2009].

2.3.1.2 Geometry-based method

Geometry-based methods focus on identifying pockets on the protein surface that can accommodate small ligand molecules by computing some types of geometric measurements. Statistically, studies of protein-ligand complex structures from the Protein Data Base have shown that small molecule ligands tend to bind to deflated regions of the protein surface, particularly its largest and/or deeper cavities [Kukol, 2014]. Typically, the first step in this category is to identify an empty space on the surface of the protein. The next step is to group the empty spaces to identify the largest pocket or cavity, which is often assigned as the best-predicted ligand binding site [Kukol, 2014]. LIGSITE^{CSC} [B. Huang & Schroeder, 2006], PocketPicker [Weisel et al., 2007], VICE [Tripathi & Kellogg, 2010], SCREEN [Nayal & Honig, 2006], POCASA [Yu et al., 2010], CASTp [Binkowski et al., 2003], MSPocket [Zhu & Pisabarro, 2011], and fpocket [Le et al., 2009] are all examples of geometry-based method.

2.3.1.3 Energy-based method

Energy-based methods aim to find patches on the protein surface that are favorable for ligand binding [Kukol, 2014]. For this purpose, a probe molecule is designed, and the interaction energy between the surrounding protein atoms and the probe is calculated [Kukol, 2014]. These

methods include SiteHound [Ghersi & Sanchez, 2009], Q-SiteFinder [Laurie & Jackson, 2005], Morita's method [Morita et al., 2008], and FTSite [Ngan et al., 2012].

2.3.1.4 Propensity-based method

The propensity-based method looks at the statistics of certain properties to determine their propensity to be at or associated with known ligand-binding sites [Kukol, 2014]. These methods often re-rank pockets predicted by other methods by finding statistically significant differences between the ligand binding site and the non-ligand binding site [Kukol, 2014]. Propensity-based methods include STP [Mehio et al., 2010], LISE [Xie & Hwang, 2012; Xie et al., 2013], and Hirayama's method [Soga et al., 2007].

2.3.1.5 Combination-based and others

These methods combine two or more methods to predict the ligand binding site. The idea is that, for example, geometry and energy are two distinct properties of ligand binding sites, and since different methods can complement or cancel each other out, it is not surprising that a combination of these methods can be successful in predicting ligand binding sites [Kukol, 2014]. Examples include ConCavity [Capra et al., 2009], MEDock [Chang et al., 2005], Thornton's method [Gutteridge et al., 2003], MetaPocket 2.0 [Huang, 2009] and P2Rank [Krivák & Hoksza, 2018]. MetaPocket 2.0 reports the consensus of the results of 8 different methods [Huang, 2009; Kukol, 2014]. The P2Rank algorithm is based on the classification of points evenly distributed on the protein's Solvent Accessible Surface (SAS points). Initially, SAS points are described by a vector of Physico-chemical, geometric, and statistical properties computed from their local geometric neighborhood. A machine learning-based model then assigns a predicted ligandability score to each SAS point. Finally, the points with high predicted ligandability – ligand binding capacity, scores are clustered to form predicted ligand binding sites [Krivák & Hoksza, 2018].

2.3.2 Binding pocket comparison

Analyses of protein-ligand complexes deposited in the Protein Data Bank (PDB) have shown that most small organic molecules interact with specific pocket-shaped indentations on the surface of their target proteins [Govindaraj & Brylinski, 2018]. These surface regions are called binding sites or binding pockets [Govindaraj & Brylinski, 2018]. It is now well known that distant proteins can have comparable binding sites with the ability to recognize chemically similar ligands [Govindaraj & Brylinski, 2018]. Various computational tools have been developed to assess the similarity of binding sites in proteins. These include PocketMatch (a web server that generates a score, PMScore for a pair of pockets compared) [Yeturu & Chandra, 2008]; eF-seek and eF-site (a web server that searches for similar ligand binding sites) [Kinoshita et al., 2007; Kinoshita et al., 2002]; Patch-Surfer: a web server that uses the 3 Dimensional Zernike Descriptor (3DZD) and Approximate Patch Position (APP) to describe the features of different patches of the protein pocket, then retrieves similar pockets in the pocket database based on the similarity of 3DZD and APP between patches of different pockets, to predict the binding ligand [Sael & Kihara, 2012]; and CavBase: uses graph theory and clique detection algorithms [Schmitt et al., 2002] to capture similarities between binding sites [Govindaraj & Brylinski, 2018].

The alignment-based methods use tools like SiteEngine: a web server that recognizes regions on the surface of one protein that resembles a specific binding site of another [Shulman-Peleg et al., 2005]; Alignment of Pockets (APoc): uses iterative dynamic programming and integer programming to determine the best alignment between two binding sites while taking secondary structure and fragment fitting into account [Gao & Skolnick, 2013]; SOIPPA (Sequence Order-Independent Profile-Profile Alignment): an algorithm capable of detecting a priori local similarity between unknown binding sites and employing a scoring function that integrates geometric, evolutionary, and physical information into a unified framework [Xie & Bourne, 2008]; and Graph-based Local Structure Alignment (GLoSA): aligns protein local structures in a sequence-independent manner and generates a GA-score, a size-independent quantity of structural similarity

for a given pair of local structures [H. S. Lee & Im, 2016, 2017]. GLoSA is a standalone program, whereas APoc has both a web server and a standalone version that can be used to compare multiple pockets in a single run.

2.3.3 GPCR ligand binding prediction

A major advance in the study of GPCRs as drug targets is the identification of ligands that bind to GPCRs [Seo et al., 2018]. Many biochemical or bioinformatic approaches have been proposed to identify drug binding to the receptor, with a focus on the calculation of protein-ligand binding affinity, which relies heavily on 3D structures of proteins or ligands [Seo et al., 2018]. Current computational databases contain a large number of molecules and are freely accessible, making computational ligand discovery a compelling alternative to high-throughput screening and other experimental approaches in the early stages of ligand discovery [Raschka & Kaufman, 2020].

There are two main approaches to computational ligand discovery. These are the ligand-based virtual screening and the structure-based virtual screening [Raschka & Kaufman, 2020]. Ligand-based virtual screening approaches focus on the structure and physicochemical properties of ligands in the absence of a receptor structure, while structure-based virtual screening requires knowledge of the receptor structure, such as molecular docking [Raschka & Kaufman, 2020]. Structure-based virtual screening uses scoring functions to identify favorable ligand candidates for the protein. These scoring functions fall into four classes: force field, empirical, knowledge-based, and machine learning-based [Raschka & Kaufman, 2020].

In recent years, machine learning has improve both ligand-based virtual screening and structure-based virtual screening [Li et al., 2019; Raschka & Kaufman, 2020]. Techniques used include random forest, convolutional neural network, deep neural network, decision tree and support vector machine [Seo et al., 2018; Li et al., 2019; Raschka & Kaufman, 2020]. Some of these techniques are based on a set of descriptors derived from the proteins and ligands. Descriptors comprise the properties of atoms in protein-ligand complexes [Hassan, 2018; Jiménez et al.,

2017], ligand structures and sequences of GPCR amino acid motifs [Seo et al., 2018]. These descriptors include the following but are not limited to [Seo et al., 2018; Hassan, 2018; Jiménez et al., 2017]:

- Hydrophobic: amino acid repels or fails to mix with water
- Aromatic: an amino acid that includes an aromatic ring
- Hydrogen bond acceptor: the amino acid of a hydrogen bond which does not supply the bridging (shared) hydrogen
- Hydrogen bond donor: the amino acid of a hydrogen bond that does supply the bridging (shared) hydrogen
- Positive ionizable (Gasteiger positive charge): ability to form ionic bonds and become positively charged
- Negative ionizable (Gasteiger negative charge): ability to form ionic bonds and become negatively charged
- Metallic (Mg, Zn, Mn, Ca, or Fe): the ability to bind to metals

These descriptors are measured either on a scale or based on counts: thus, the number of amino acids that are hydrophobic, aromatic, hydrogen bond acceptor or donor etc. How many can bind to metal ions?

2.3.4 Protein-ligand docking

The goal of protein-ligand docking is to predict the position and conformation of a ligand when it binds to a protein receptor or an enzyme [Taylor et al., 2002; Forli et al., 2016]. Several protein-ligand docking applications are available, such as, AutoDock and AutoDock Vina [Morris et al., 2009; Trott & Olson, 2009], rDock [Ruiz-Carmona et al., 2014], Surflex [Spitzer & Jain, 2012] and Glide [Friesner et al., 2004, 2006]. These applications calculate the location, geometry, and energy of ligands or peptides interacting with proteins. They have been used to couple GPCRs

to their associated ligands. There are some that are specific to GPCRs, such as, IPHoLD, an integrated protein homology model, ligand docking and protein design approach that models conformational selection and ligand binding modes with induced matching only in terms of homologous receptor structures [Feng et al., 2017]. Figure 4 shows an illustration of protein-ligand docking.

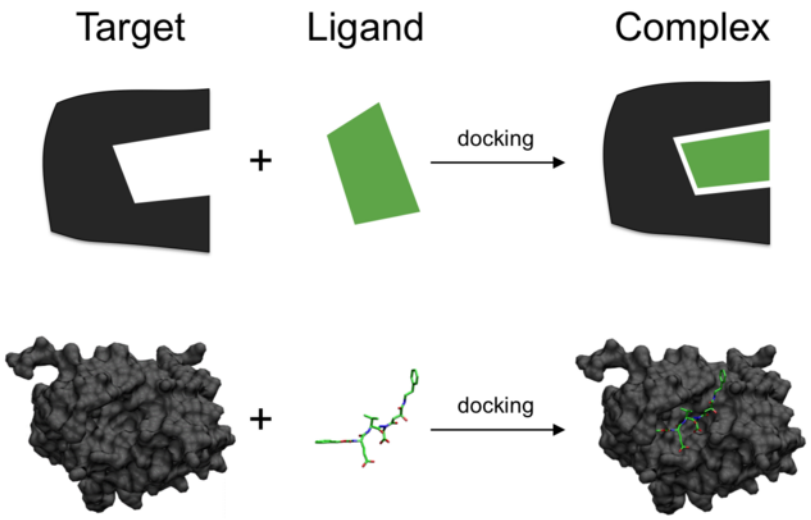


Figure 4: Schematic illustration of docking a small molecule ligand (green) to a protein target (black) producing a stable complex (source: Wikipedia: https://en.wikipedia.org/wiki/Docking_(molecular))

Structural similarities in binding poses between small molecules have been shown to be important in docking. RosettaLigandEnsemble (RLE) was developed based on structural similarities in binding poses among small molecules that bind to one binding pocket [Fu & Meiler, 2018]. RLE has been found to generate more consistent docking results within a congeneric series and salvage failed docking of individual ligands [Fu & Meiler, 2018]. Malhotra & Karanicolas, (2017) showed that for the 14% of pairs of related ligands that are resolved in complex with the same partner protein, their mode of binding changes with chemical manipulation of the smaller ligand in the pair. They have shown that simple structure-based modeling is more effective for

identifying chemical substitutions that alter the binding mode for these pairs of ligands [Malhotra & Karanicolas, 2017]. Some ligand pairs change binding mode because the added substituent would irreconcilably conflict with the receptor in the original pose, whereas others change because the added substituent enables new, stronger interactions that are available only in a different pose [Malhotra & Karanicolas, 2017].

2.4 BINDING POCKET ELECTROSTATIC PROPERTIES

The electrostatic properties of proteins are derived from the proportion and distribution of polar and charged residues [Sinha & Smith-Gill, 2002]. Polar and charged residues use electrostatic properties to form short-range interactions such as salt bridges and hydrogen bonds and defining the overall electrostatic environment in the protein [Sinha & Smith-Gill, 2002]. Short-range and long-range electrostatic interactions, along with other forces, provide guidance cues in molecular and macromolecular assembly [Vascon et al., 2020]. Charged and polar groups impart important properties to proteins through ion pairing, hydrogen bonding, and other less specific electrostatic interactions [Zhou & Pang, 2018]. Electrostatics plays an important role in defining the mechanisms of protein-protein complex formation, molecular recognition, thermal stability, conformational adaptability, and protein movement [Sinha & Smith-Gill, 2002]. For example, increased binding specificity and affinity involve optimization of electrostatics; high-affinity antibodies have higher and stronger electrostatic interactions with their antigens; the rigid parts of proteins have higher and stronger electrostatic interactions [Sinha & Smith-Gill, 2002].

Using the R package *Peptides* [Daniel Osorio, 2021], we calculated Molecular Surface Weighted Holistic Invariant Molecular (MS-WHIM) scores (consisting of three values, x, y, and z) of the amino acids in each pocket [Bravi et al., 1997; Zaliani & Gancia, 1999]. MS-WHIM

scores are calculated using electrostatic potential properties derived from the 3D structure of the 20 natural amino acids, as described in the references [Bravi et al., 1997; Zaliani & Gancia, 1999] To visualize the electrostatic properties, the electrostatic surface properties of the pockets were generated using the PyMOL plugins APBS and PDB2PQR [Jurrus et al., 2018].

2.5 PROTEIN STRUCTURAL COMPARISON

Comparative analyzes of protein sequences and structures play a central role in understanding proteins and their functions [Hasegawa & Holm, 2009]. Assuming an evolutionary continuity of structure and function, describing the structural similarity relationships among protein structures permits scientists to deduce the functions of newly found proteins [Holm & Laakso, 2016]. Several protein structure comparison programs have been developed, such as CE [Shindyalov & Bourne, 1998], DALI [Holm & Sander, 1993], MultiProt [Shatsky et al., 2002], and FATCAT [Ye & Godzik, 2003; Ye & Godzik, 2004]. Some of these programs, for example, FATCAT, take flexibility and structural rearrangements of proteins into account. FATCAT optimizes alignment and minimizes the amount of rigid body motion (twists) about pivots (hinges) introduced into the reference protein [Ye & Godzik, 2003; Ye & Godzik, 2004]. FATCAT generates a structural similarity score (RMSD), sequence similarity, and identity score for each pair of GPCRs it compares. Figure 5 shows a flexible and a rigid structural alignment of GPCRs with PDB IDs 3PBL and 6CM4.

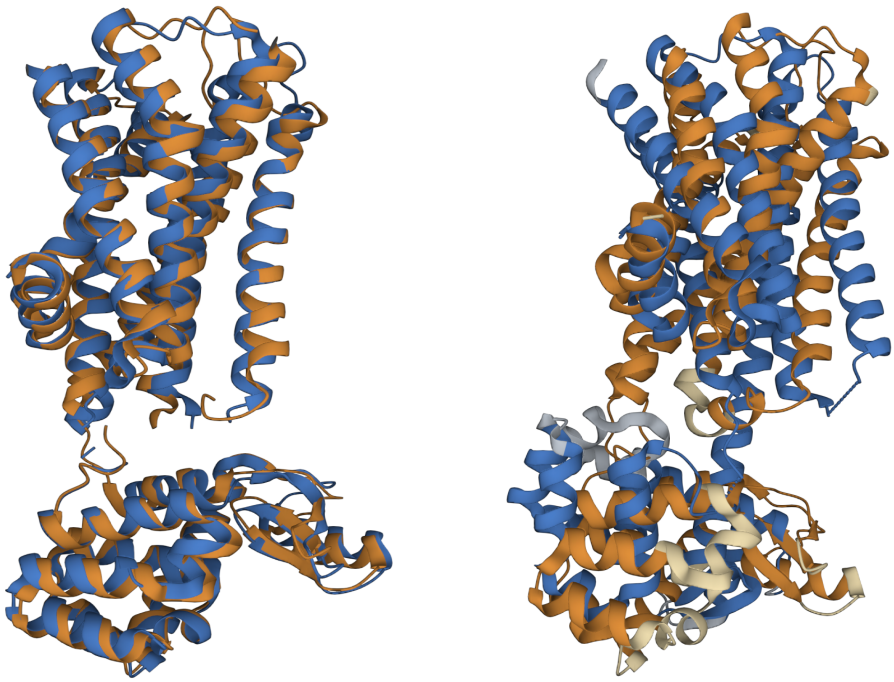


Figure 5: FATCAT Structural alignment of PDB: 3PBL and PDB: 6CM4 (Left: Flexible, right: Rigid)

Chapter 3: Materials and Methods

In this chapter we first describe the construction of our datasets: GPCR ligand interaction and 3D structures. This is followed by a description of methods of motif search, 3D structural comparisons, binding pockets prediction and comparison, GPCR ligand docking, ligand binding pose and conformation, protein-ligand interaction, pockets, and 3D structural similarity, and pocket electrostatic properties. Appendix B contains detailed step-by-step instructions that one can follow to complete the analysis. Figure 6 shows the workflow of this study. Appendices C through G contain the Python and R codes developed for the various steps of the workflow. A copy is available at github.com/owusukd/GPCR_Ligand_Interaction.

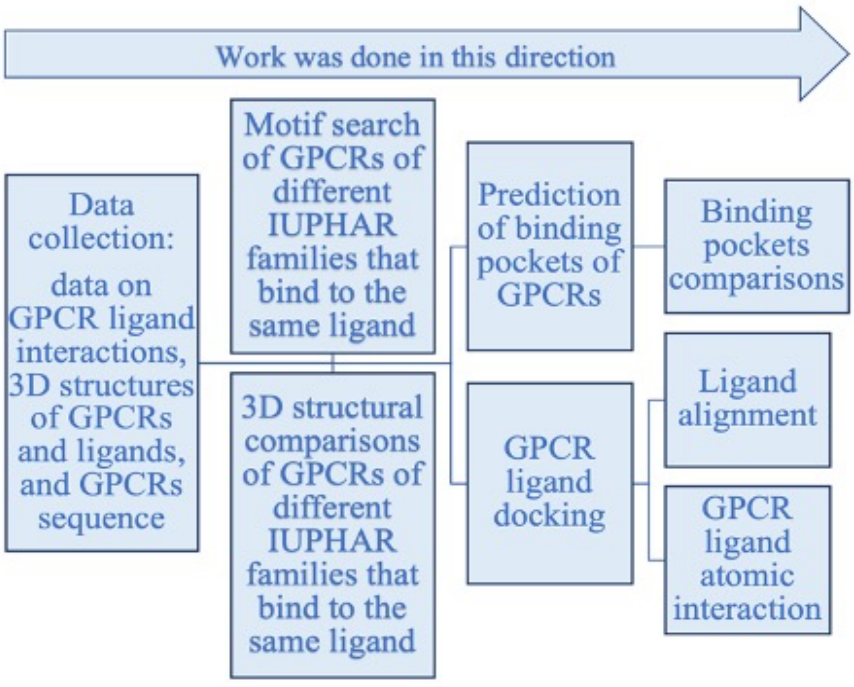


Figure 6: Workflow overview

3.1 Dataset collection

We compiled publicly available datasets on GPCR 3D structures and sequences for structural comparison, molecular docking, and sequence motif search. In addition, for the GPCR

sequences, we collected information on the beginning and end of their regional sequences (the N-terminal, extracellular loops, intracellular loops, the seven trans-membrane helices, and the C-terminal).

Beginning with confirmed GPCRs in the GPCR-PEnDB database [Begum et al., 2020] information on GPCR ligand interactions was gathered, compiled, and restructured to aid computational analysis and to identify ligands that bind to GPCRs in three different IUPHAR [Armstrong et al., 2020] families.

Data on interactions were gathered from IUPHAR, BindingDB [Gilson et al., 2016], and GLASS [Chan et al., 2015]. It also includes data on each ligand's SMILES (Simplified Molecular Input Line Entry Specification—a linear notation for describing chemical structures), affinity relations, InChIKey (International Chemical Identifier compact hashed code), potency, activity, inhibition, affinity, and action (i.e., agonist, full agonist, partial agonist, antagonist, inverse agonist, biased agonist, etc.).

As positive controls for the various docking analyses (sections 4.5 & 4.6), 11 ligands that bind to a mix of GPCRs from the same family, different families, and non-GPCRs with protein-ligand complexes on PDB (see Section 4.2: Table 5) were gathered. The 11 ligands were chosen to have GPCR representations from various IUPHAR families. However, due to a lack of GPCR ligand complexes on PDB, not all families had representations.

3.1.1 Data Sources

Five datasets have been collected and saved. Below is a description of the datasets:

1. Research Collaboratory for Structural Bioinformatics Protein Data Bank (PDB) dataset contains available 3D structural data on GPCR. This includes GPCR PBD IDs as well as experimental structural factor data (this includes information about the atoms, coordinates of the atoms in space, etc., that makes up the 3D structure of the protein).

2. BindingDB dataset contains information on the ligands and the GPCRs they bind to. This information includes binding affinities, ligand molecular weight, and SMILES (Simplified Molecular Input Line Entry System).
3. GLASS (GPCR-Ligand ASSociation) had data on the ligands that bind to GPCRs. These data include ligand IDs and binding affinities.
4. IUPHAR/BPS dataset contained information on the actions of ligands on GPCRs, such as whether they are agonists, full agonists, partial agonists, antagonists, inverse agonist, biased agonists, neutral, positive, or negative.
5. Finally, we obtained the confirmed GPCR sequences from GPCR-PEnDB (GPCR Prediction Ensemble Database; gpcr.utep.edu/database) and UniProtKB. In addition, we obtained data from UniProtKB on the beginning and end of the GPCRs' regional sequence (the N-terminal, extracellular loops, intracellular loops, seven helices, and C-terminus).

The GPCR sequence data was saved as a single fasta file. We saved their PDB ID as a single csv file for the 3D structural data, and then their experimental data on their structural factors as a pdb file. A text file containing data on the beginning and end of the regional sequence was created. All other data was saved as a single csv file.

3.1.2 Procedures for data acquisition

We obtained a list of UniProt IDs of confirmed GPCRs from our database (originally from UniProtKB), GPCR-PEnDB, and using this list, and we searched PDB for each of them to see if there was a 3D structure for it, using the advanced search available on PDB. With the custom table option, we selected the PDB ID, experimental method, ligand ID, and Accession code(s) from the search results and downloaded the resulting csv file. We discovered that the column in the data for UniProt ID contained multiple UniProt IDs after performing some exploratory data analysis (on each column/variable). As a result, we took that column and cross-checked it against the list of confirmed GPCRs to eliminate any non-GPCRs from the list. After that, we crosschecked the

elimination result to see if there was a PDB entry for each of the UniProt IDs we discovered. Following confirmation of the result, we use the advanced search to search and download data on each at a time, avoiding the situation described above. We then downloaded information about their 3D structure. GPCR-PEnDB was used to obtain the protein sequences of the confirmed GPCRs (originally from UniProtKB). Also, data on the start and end of the regional sequence (the N-terminal, extracellular loops, intracellular loops, the seven helices, and the C-terminal) of the GPCRs were obtained from UniProtKB.

To obtain data from BindingDB, we had to create an account with them to get access to the data. We searched for each of the confirmed GPCR on their site using the UniProt IDs of the GPCRs. After getting a hit, we had to enter our login credentials and security code generated by the site, each time we wanted to download data on the hit. The data came in a tsv (tab-separated values) file.

In the case of IUPHAR/BPS, we navigated within their website to "download data and reports" through "downloads," which was under "resources." On the "download data and reports" page, we downloaded data on "complete ligand list", and "all interaction data for ligands and targets," which were listed under the subheading "Download data files". In addition, the interaction data contained information on whether the ligand is an antagonist or an agonist.

3.1.3 Exploratory data analysis and data restructuring

After gathering all the necessary data mentioned above, we conducted exploratory data analysis on them. PDB IDs of GPCRs obtained from the PDB were cleaned to remove duplicates. Ligand data from BindingDB, IUPHAR/BPS, and GLASS were restructured to include all binding affinities, namely EC50, IC50, Ki, Kd, pKb, pEC50, pKi, pIC50, pKd, potency, activity (in %), and inhibition (in %) in a single column. The binding affinities were cleaned of relational symbols (>, etc.) and reorganized into a new column. We merged ligand data from BindingDB, IUPHAR/BPS, and GLASS based on GPCR IDs after restructuring. The resulting data was

duplicate-free and saved as a csv file. The exploratory data analysis and data restructuring were carried out using custom scripts I wrote in the R programming language (Appendix C.1 to C.5).

3.2 MOTIF SEARCH

During the exploratory data analysis, we discovered that some ligands bound to GPCRs from two or more distinct IUPHAR families. As a result, we decided to conduct a motif search across the GPCRs that these ligands bound to to see if any significant motifs traverse GPCRs IUPHAR families. We concentrated on ligands that bound to GPCRs from three different IUPHAR families. We searched for motifs using the Multiple Expression Motifs for Motif Elicitation (MEME) system. MEME searches for repeated, un-gapped sequence patterns in DNA or protein sequences provided by the user [Bailey et al., 2006; Bailey et al., 2009; Bailey et al., 2015]. In our motif search, we took three approaches:

1. Perform a motif search on the entire sequence of the GPCRs.
2. Perform a motif search on the regional sequence (the N-terminal, extracellular loops, intracellular loops, the seven helices, and the C-terminal) of the GPCRs.
3. Perform a motif search on the modified regional sequence (that is start the regional sequence five amino acids before the actual start of the regional sequence and/or end the regional sequence five amino acids after the actual end of the regional sequence) of the GPCRs.

To conduct these motif searches, we obtained the start and end of the regional sequence (the N-terminal, extracellular loops, intracellular loops, seven helices, and C-terminal) of the GPCRs from UniProt. These data were used to divide the GPCR sequence into different regions (N-terminal, extracellular loops, intracellular loops, seven helices, and C-terminus) using the R scripts and resulting sequence regions were saved as fasta files. After segmenting the sequences into regions, we used MEME to search for motifs. Motifs with an E-value < 0.1 were kept as

significant motifs. All sequence cutting and motif searches were carried out using R and shell scripts, respectively (Appendix D).

3.3 STRUCTURAL COMPARISON

In our quest to understand why some ligands bind to GPCRs of three distinct IUPHAR families, we decided to investigate whether there are some 3D structural similarities across different IUPHAR families that facilitate the binding. To achieve this, we performed pairwise 3D structural comparisons of the GPCRs bound to the same ligand and accessed the comparisons based on the alignment's RMSD (root mean square deviation) value of the alignment. The comparisons were made using Flexible structure AlignmenT by Chaining Aligned fragment pairs allowing Twists (FATCAT) [Ye & Godzik, 2003; Ye & Godzik, 2004]. FATCAT starts by identifying a list of AFPs (aligned fragment pairs)—a superposition of two continuous fragments—in the two proteins to be compared [Ye & Godzik, 2003; Ye & Godzik, 2004]. The FATCAT structure alignment is formulated as an AFP chaining process that allows for flexibility in connecting them, that is, introducing between two consecutive AFPs a rotation/translation (twist) if it results in a substantially better superposition of the structures [Ye & Godzik, 2003; Ye & Godzik, 2004].

We made use of the stand-alone version of FATCAT, which is publicly available on GitHub (https://github.com/GodzikLab/FATCAT-dist). We performed a rigid FATCAT and a flexible FATCAT. The rigid FATCAT uses a rigid-body superposition to align the two structures, whereas the flexible FATCAT introduces 'twists' between different parts of the proteins, which are superimposed independently [Prlić et al., 2010; Ye & Godzik, 2004; Ye & Godzik, 2003]. We wrote Python code to map the sequence alignment from FATCAT to the GPCR structures under comparison in order to extract the portions of their 3D structures that were found to be structurally similar. These portions were used for further analysis (see Section 3.9). For GPCRs that had multiple structures deposited in PDB, we selected one of their PDB IDs with the longest protein

sequence of the GPCR. In other words, the one with a protein sequence length closer to the actual GPCR sequence length was chosen for 3D structural comparison. The 3D structural comparisons were carried out using shell script and python codes (Appendix E.1).

3.4 BINDING POCKET PREDICTION

Three-dimensional structures of the GPCRs under consideration were downloaded from PDB and cleaned of any unwanted molecules, including ligands not under study in this work and molecules used to aid the determination of the 3D structure of the GPCRs. If the binding site of the ligand is known, we compare it with the binding site or predicted pockets of the other GPCRs that bind to the same ligand. If the binding site of the ligand is unknown, we submit the cleaned 3D structure to the P2Rank (version 2.2) stand-alone version for binding pocket prediction. None of the ligands had a known binding site for the GPCRs they interact with, so we submitted all the GPCRs they interact with to P2Rank. Results from P2Rank come in the form of a csv file containing the pocket number, pocket center in xyz-coordinates, and position IDs of the amino acids that form the pocket. As a result, we wrote a Python code (Appendix E.2.1.3) to extract the predicted pockets from the 3D structure of the GPCR using the position IDs from the pocket prediction results. Finally, all predicted binding pockets were considered for pocket comparison and for molecular docking. Binding pocket predictions were carried out using shell script (Appendix E.2.1.1 & E.2.1.2).

3.5 BINDING POCKET COMPARISON

All binding pockets predicted by P2Rank for each of the GPCRs were compared against those of the GPCRs they share the same ligand with. The comparisons were performed using APoc [Teilum et al., 2009] which implements iterative dynamic programming and integer programming to calculate the optimal alignment between a pair of binding sites considering the secondary structure and fragment fitting. APoc provides a scoring function called the Pocket Similarity (PS)-

score, which quantifies the pocket similarity between two given pockets based on their backbone geometry, the orientation of side chains, and chemical matching of aligned pocket residues [Teilum et al., 2009;Schmitt et al., 2002]. APoc can be applied to both experimentally determined and computationally predicted ligand binding sites [Shulman-Peleg et al., 2005]. The scoring function of APoc takes into consideration the chemical similarity of the aligned amino acids of the pockets in comparison [Shulman-Peleg et al., 2005]; as a result a separate chemical similarity score of the predicted binding residues of the pockets in contrast was not necessary. APoc was chosen over others for its good performance and ease of comparing multiple pockets in one run. Binding pocket comparisons were carried out using shell script and results were gathered using python code (Appendix E.2.2).

3.6 GPCR LIGAND DOCKING

Protein-ligand interactions are generally specific, in terms of the ligand involved, and the location the interaction takes place [Kukol, 2014]. As a result of the location specificity of the ligand binding sites, we decided to perform molecular docking between the ligands that bind to GPCRs of three distinct IUPHAR families to find out what they have in common at the binding sites and what binding residues they share at the binding sites. In so doing we downloaded 3D structures of these GPCRs from PDB and were prepared using a shell script. The preparation involved deleting any metal compounds and water molecules, deleting ligands, and adding and optimizing hydrogen bonds. We also added Gasteiger partial charges to the protein. The prepared protein was saved having the file extension pdbqt. Gasteiger partial charges are commonly referred to as net atomic charges that are used in molecular mechanics force fields to compute the electrostatic interaction energy [Kramer et al., 2014]

We converted the SMILES (Simplified Molecular Input Line Entry System) of the ligands into 3D structures using Open Babel [O'Boyle et al., 2011] adding and optimizing hydrogen bonds.

We also added Gasteiger partial charges to the ligand and saved them as pdbqt files using Open Babel.

The ligands were docked using AutoDock Vina [Trott & Olson, 2009] into all the predicted binding pockets for the GPCRs they are known to bind to. This was done because at the time of this study there were no known 3D complexes of the ligands bound to the GPCRs on PDB [Berman et al., 2002]; therefore, we had no information of where and how the ligand binds to the GPCRs. Six ligand modes (poses and conformations) were generated, and only the ligand mode with the most negative value for the binding affinities (i.e., strongest binding) were retained for further analysis. We assessed the correlation between the PS-scores provided by APoc and the absolute difference of the binding affinities. Note that the smaller this absolute difference, the more alike the two pockets are in terms of their binding strengths for the same ligand. A negative correlation would imply that the PS-score is a good predictor of similarity in binding affinities.

3.7 LIGAND BINDING POSE AND CONFORMATION

After docking the ligands to their respective GPCRs, we aligned the docked ligand 3D structures using the PyMOL structural alignment method align, "which" performs a sequence alignment followed by a structural superposition. This allowed us to assess the conformation of the docked ligands using the root mean squared deviation (RMSD) from the alignment. The Pearson correlation between the pocket similarity score (PS-Score) and the RMSD was assessed. The ligand poses were visually examined using PyMOL.

To assess the reliability of the docking results of the study data, we aligned the ligands of the pairs of pockets as deposited on PDB and recorded the RMSD from the alignment, $RMSD_{Actual}$, of the control data. We then docked the same ligands into the same pockets as found on PDB and then aligned the ligands docked into the pairs of pockets of the control data. The RMSD from the alignment of the docked ligands were recorded as $RMSD_{Docked}$. We assessed the reliability of the

docking results by measuring the difference between the two sets of RMSD data of the control data and the relationship between PS-scores and $RMSD_{Actual}$.

3.8 PROTEIN LIGAND INTERACTION

After the ligands have been docked into all the predicted binding pockets, we use LigPlot+ [Laskowski & Swindells, 2011] to determine which residues of the GPCRs interact with which atoms of the ligand. LigPlot+ generates schematic 2D representations of protein–ligand complexes with colored outputs, postscript files containing information on intermolecular interactions and their strengths, including hydrogen bonds, hydrophobic interactions, and atom accessibilities [Laskowski & Swindells, 2011]. LigPlot+ was used to ascertain the interaction of retinal with the lysine amino acid of rhodopsin as proposed by Malhotra & Karanicolas, (2017). This interaction was detected within a threshold of 5Å. Based on this, we chose hydrogen bonds detected within 5 Å by LigPlot+. We gathered data on the number of same residues across the pockets we are comparing that interact with the ligand from the LigPlot+ results output.

3.9 PREDICTED POCKET AND 3D STRUCTURAL SIMILARITY COMPARISON OVERLAP

After the GPCRs that bind the same ligand have been compared based on their 3D structure, we compared the portions of the proteins that were similar from the pairwise 3D structure comparison (see Section 3.3) to their own predicted pockets. For example, let the 3D structures of the GPCR with PDB ID 3G04 be A, and that of the GPCR with PDB ID 7LCK be B. Let P_A be the part of A that was found to be 3D structurally similar to B, likewise, P_B be the part of B that was found to be 3D structurally similar to A. Let $PK_{(A,i)}$, $i = 1, 2, \ldots, m$ be the predicted pockets of A and $PK_{(B,j)}$, $j = 1, 2, \ldots, n$ be the predicted pockets of B. Each $PK_{(A,i)}$, $i = 1, 2, \ldots, m$ was compared with P_A and each $PK_{(B,j)}$, $j = 1, 2, \ldots, n$ was compared with P_B. An "overlap score" was calculated for each comparison using the formula below:

$$S_{(A,i),B} = \frac{N(aaP_A \cap aaPK_{(A,i)})}{N(aaPK_{(A,i)})},$$
$$S_{(B,j),A} = \frac{N(aaP_B \cap aaPK_{(B,j)})}{N(aaPK_{(B,j)})}, \quad (1)$$

where $S_{(A,i),B}$ is the overlap score for the ith pocket of A with P_A when A is compared with B, $S_{(B,j),A}$ is the overlap score for the jth pocket of B with P_B when B is compared with A. In equation (1), aaP_A, aaP_B, $aaPK_{(A,i)}$, and $aaPK_{(B,j)}$ respectively denote the sets of ordered positions of amino acids in P_A, P_B, $aaPK_{(A,i)}$, and $aaPK_{(B,j)}$, and $N(\bullet)$ represents the count of the amino acid positions. The code for the overlap scoring is included Appendix E.3. Overlap scores were calculated for both cases of using flexible 3D structural comparisons and rigid 3D structural comparisons (see Section 3.3). An average score for the flexible and the rigid cases were obtained for each pocket. The average scores were summed together for each pair of pockets compared using the formula below:

$$S_{(A,i),(B,j)} = \bar{S}_{(A,i),B} + \bar{S}_{(B,j),A}, \quad (2)$$

where $\bar{S}_{(A,i),B}$ and $\bar{S}_{(B,j),A}$ are the average overlap score for the flexible and the rigid case of pocket $PK_{(A,i)}$ and $PK_{(B,j)}$, and $S_{(A,i),(B,j)}$ is the sum of average overlap scores for the pair of pockets, $PK_{(A,i)}$ and $PK_{(B,j)}$, being compared. The summed average overlap scores obtained by Equation (2) were then analyzed to determine their Pearson correlation with the PS-scores. This was done to check if the parts of the proteins that are 3D structurally similar are also locations for pockets.

3.10 POCKETS ELECTROSTATIC PROPERTIES

We calculated Molecular Surface Weighted Holistic Invariant Molecular (MS-WHIM) scores (made up of three values, x, y, z) [Bravi et al., 1997; Zaliani & Gancia, 1999] of the amino acids of each of the pockets using the R package *Peptides* [Daniel Osorio, 2021]. MS-WHIM scores are obtained from electrostatic potential properties derived from the 3D structure of the 20 natural amino acids as described in references [Bravi et al., 1997; Zaliani & Gancia, 1999]. We calculated the Chebyshev distance between the MS-WHIM scores of any two pockets under comparison. For example, let $PK_{(A,i)}$, $i = 1, 2, \ldots, m$ be the predicted pockets of GPCR A and $PK_{(B,j)}$,

$j = 1, 2, \ldots, n$ be the predicted pockets of GPCR B that are under comparison. For each pocket $PK_{(A,i)}$ and $PK_{(B,j)}$, we calculate MS-WHIM scores of the amino acids that form the pockets. These scores $(x, y, z)_{PK_{(A,i)}}$ and $(x, y, z)_{PK_{(B,j)}}$ for the pockets we are comparing were seen as points in the 3D-space, and the distance D between the two points was calculated using Chebyshev distance.

$$D\left[(x, y, z)_{PK_{(A,i)}}, (x, y, z)_{PK_{(B,j)}}\right] = \max\left(|x_i - x_j|, |y_i - y_j|, |z_i - z_j|\right). \quad (3)$$

The Chebyshev distance was calculated for each pair of pockets under comparison. The distances were later used to determine the Pearson correlation between the PS-score and MS-WHIM. To visualize the electrostatic properties, electrostatic surface properties of the pockets were generated using APBS and PDB2PQR [Jurrus et al., 2018] plugins for PyMOL.

Chapter 4: Results and Discussion

In this chapter we have the overall results on all the methods discussed above.

4.1 Currently Known 3D Structures of GPCR

Below are figures of the frequency distribution of the 3D structures we have collected. There were 817 atomic-level 3D GPCR structures related to 161 distinct GPCRs, of which 107, 27, 16, 2, 0, 9, and 0 are in Class A, B, C, D, E, F, and T2R, respectively, on PDB (June 2022) (see Figure 7). There were no 3D structures for Class E, however, Class A GPCRs had the most 3D structures on PDB. Figure 8 shows the frequency distribution by year. Generally, the number of 3D structures of GPCRs is increasing rapidly by the year. We have added links from GPCR-PEnDB to the PDB structure files.

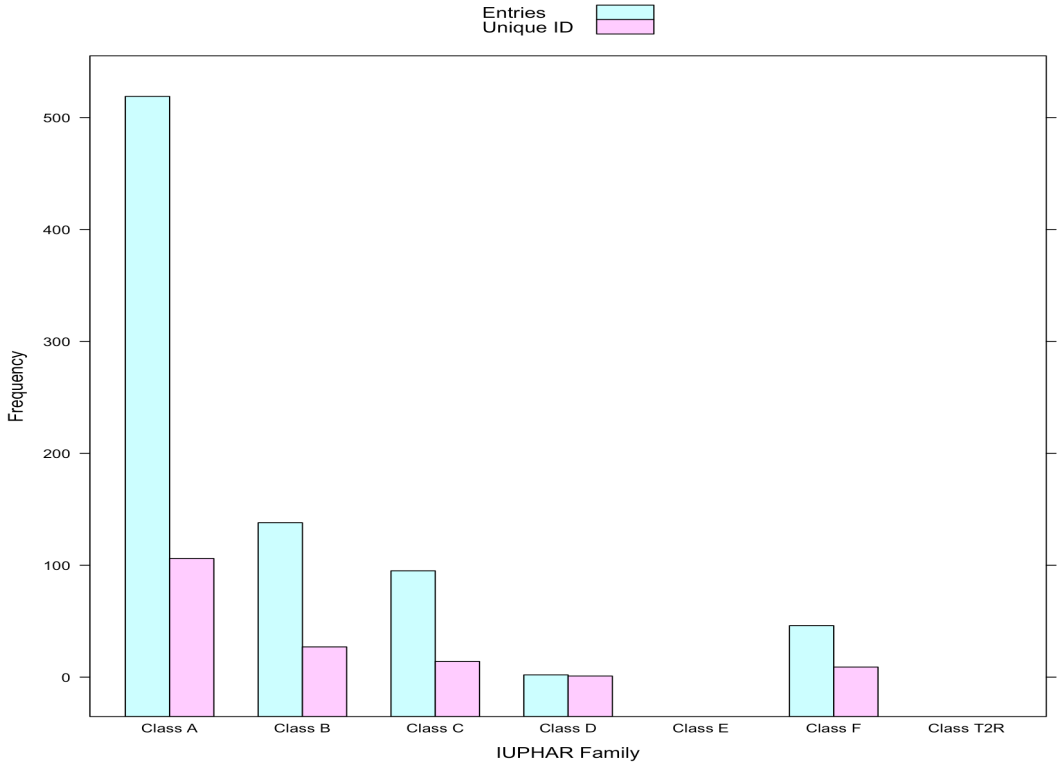


Figure 7: Frequency distribution of GPCRs 3D structures in PDB classified by family

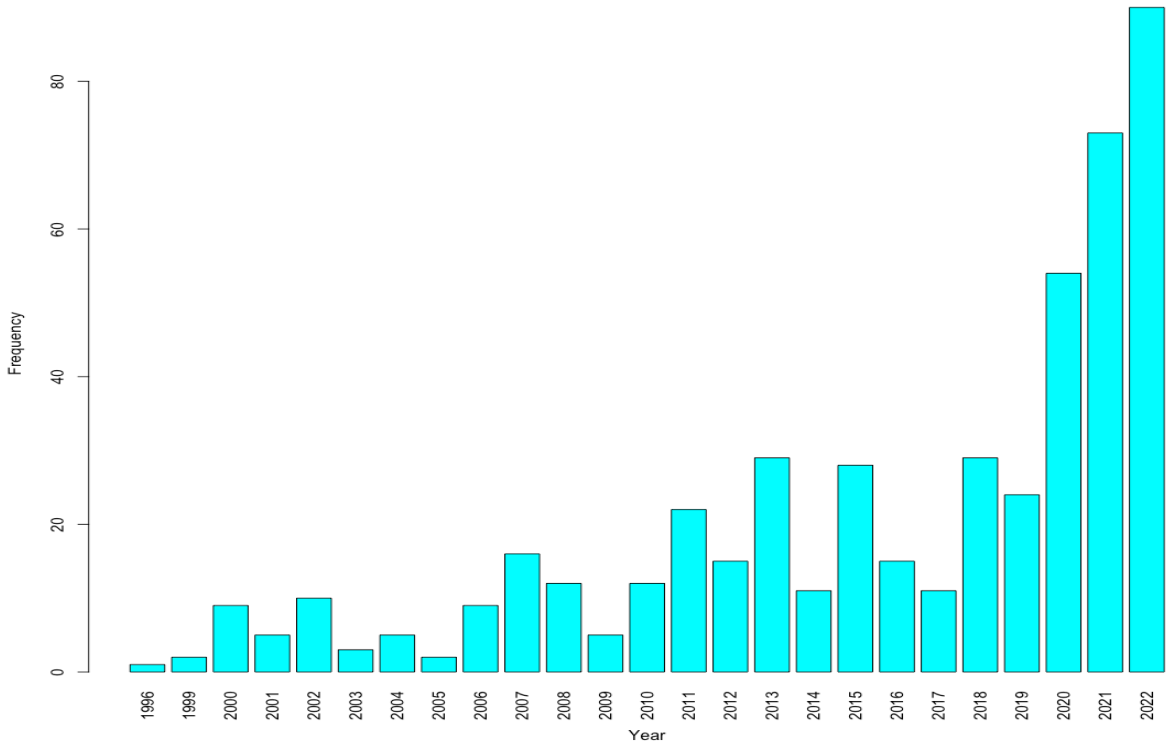


Figure 8: Frequency distribution of new GPCR 3D structures added by year

In GPCR-PEnDB we collected data on GPCR ligand interactions which was obtained from IUPHAR, BindingDB, and GLASS. The data contains a total of 1,061,462 GPCR ligand interactions with information on ligands and the binding affinities (Ki, Kd, IC50, EC50, pKB, pKi, pKd, pIC50, pEC50). In addition, it contains information on ligand's SMILES (Simplified Molecular Input Line Entry Specification: a linear notation for describing chemical structures), affinity relations, and ligand's InChIKey (International Chemical Identifier compact hashed code), potency, activity, inhibition, action (i.e., agonist, full agonist, partial agonist, antagonist, inverse agonist, biased agonist, neutral, positive, and negative).

4.2 GPCR LIGAND BINDING

Analysis on the GPCR-ligand binding data revealed that there are ligands that bind to human GPCRs of multiple IUPHAR families (Table 2). Table 2 shows over 16000 ligands interacting with human GPCR of two or more IUPHAR families. There were 11 ligands that binds

to human GPCRs of three different IUPHAR families (Table 3 & 4). These 11 ligands had 106 interactions with human GPCRs, involving 69 unique GPCRs. Of the 69 unique GPCRs, 42 had entries in RCSB PDB. IUPHAR classify three of these ligands as synthetic organic based on their nature (Table 3). The remaining eight ligands had no entry and classification in IUPHAR. IUPHAR classify ligands as approved, WHO, synthetic organics, metabolites, natural products, endogenous peptides, inorganics etc. There were no RCSB PDB entries of GPCR-ligand interaction of the 11 ligands that bind to human GPCRs of three different IUPHAR families.

Table 2: Frequency distribution of ligands by the number of distinct IUPHAR families they bind to.

Number of distinct IUPHAR families	**Frequency**
1	98750
2	16191
3	11

Table 3: List of the 11 ligands that bind to 3 different IUPHAR families

Ligands InChIKey	**Abbreviation**	**Compound Class**
XLWJPQQFJNGUPA-UHFFFAOYSA-N	XLWJ	Synthetic Organic
DTZDSNQYNPNCPK-UHFFFAOYSA-N	DTZD	Synthetic Organic
CLQVVBPDAXJGBV-UHFFFAOYSA-N	CLQV	Synthetic Organic
AJLFQFYMLRXVHV-UHFFFAOYSA-N	AJLF	Not known
IKSHHOBCJKJKOG-UHFFFAOYSA-N	IKSH	Not known
FQUAFMNPXPXOJE-UHFFFAOYSA-N	FQUA	Not known
MLQFOEOUNIRULR-UHFFFAOYSA-N	MLQF	Not known
YKMSTUDOGGAEJH-UHFFFAOYSA-N	YKMS	Not known
USZPQRMQYJIDII-UHFFFAOYSA-N	USZP	Not known
BYBLEWFAAKGYCD-UHFFFAOYSA-N	BYBL	Not known
NKOPNLUYOHOGFZ-UHFFFAOYSA-N	NKOP	Not known

It should be noted that out of the 11 ligands, 3 were found to bind to a rather large number of GPCRs (e.g., ligand FQUA bound to 25 Class A, 6 Class B, and 2 Class C GPCRs), each of

Table 4: Ligands that binds to GPCRs of three IUPHAR families.

Ligands InChI Key	GPCR UniProt ID	IUPHAR Family	Number in each Family
XLWJ	O75899, Q9UBS5	C	2
	P16473, P21728, Q6W5P4	A	3
	P43220, Q03431	B	2
DTZD	P21728	A	1
	P43220	B	1
	Q14831	C	1
CLQV	P21453, Q6W5P4	A	2
	P43220	B	1
	Q14833	C	1
AJLF	P16473	A	1
	P43220	B	1
	Q9NYV8	T2R	1
IKSH	P16473	A	1
	Q03431	B	1
	Q9NYV8	T2R	1
FQUA	O14842, O14843, O15552, O43194, P07550, P08172, P11229, P14416, P18825, P24530, P25103, P28222, P28223, P28335, P29274, P32245, P35367, P35368, P35372, P37288, P41145, Q5NUL3, Q8IZ08, Q8TDU9, Q99788	A	25
	P59536, P59551, Q9NYV7, Q9NYV8, Q9NYW1, Q9NYW5	T2R	6
	P47871, P48546	B	2
MLQF	O00222, P41594, Q13255, Q14416, Q14831, Q14832, Q14833	C	7
	Q03431	B	1
	Q6W5P4	A	1
YKMS	P14416, P21917, P35462	A	3
	P43220	B	1
	Q14416	C	1
USZP	P16473	A	1
	P43220	B	1
	P41594	C	1
BYBL	P08172, P08173, P08588, P08913, P0DMS8, P11229, P13945, P14416, P18089, P20309, P21452, P21554, P21728, P21917, P25021, P25100, P25103, P28223, P28335, P29274, P30542, P32245, P33032, P35372, P35462, P41143, P41145, P41595, P41968, P50052, P50406	A	31

Ligands InChI Key	GPCR UniProt ID	IUPHAR Family	Number in each Family
	O15303	C	1
	P43220	B	1
NKOP	P43220	B	1
	Q13255	C	1
	Q6W5P4	A	1

which may have several binding pockets. Those three ligands were estimated to generate over 30,000 binding pocket pairs in total. To reduce the number of pairwise pocket comparisons, we excluded the three ligands from this study and focused only on the remaining eight ligands for which a total of about 990 pocket pairs were compared to produce the results in Section 4.5 below. Also, we excluded these three ligands from analysis in Section 4.4 – 4.8. Table 5 shows the list of ligands and the proteins they bind to in the control dataset.

Table 5: List of Ligands and the Proteins they bind to for the Control Dataset

Ligands PDB ID	Proteins UniProt ID	Protein PDB ID	GPCR	GPCR Class
0HK	P08173	5DSG	Yes	A
	P11229	5CXV	Yes	A
7LD	P28223	6WGT	Yes	A
	P41595	5TVN	Yes	A
7MA	O43613	6TOD	Yes	A
	O43614	5WQC	Yes	A
8NU	P14416	6CM4	Yes	A
	P28223	6A93	Yes	A
40F	Q14416	4XAQ	Yes	C
	Q14832	4XAR	Yes	C
89F	P28222	5V54	Yes	A
	P28223	6WH4	Yes	A
ADN	P29274	2YDO	Yes	A
	P30542	6D9H	Yes	A
GGL	O00222	6BSZ	Yes	C
	Q14416	5CNI	Yes	C
	Q14832	5CNK	Yes	C
GLU	A0A173M0G2	5X2P	Yes	TR2
	E9P5T5	4IO2	No	-

	P41594	3LMK	Yes	A
	P42264	3S9E	No	-
	Q14416	7MTR	Yes	A
SRO	P08908	7E2Y	Yes	A
	P37231	3ADV	No	-
Z99	P41594	7FD9	Yes	C
	Q13255	3KS9	Yes	C
	Q14831	3MQ4	Yes	C
	Q14832	7WI6	Yes	C

4.3 CONSERVED MOTIFS

Upon the discovery that there are ligands that bind to human GPCRs of 3 different IUPHAR families, we set out to find why that happens. These led us to perform motif search on the sequences to determine whether there are some conserved motifs across different IUPHAR families. We performed this analysis on the full sequence, regions of the sequence (extracellular loops, intracellular loops, and the seven transmembrane helices), and modified regions of the sequence (i.e., adding 5 amino acids either at the beginning or at the end of the regional sequence or at both ends: this was done only to the extracellular loops). The regions of the GPCR sequences were labeled E_i, $i = 1, 2, 3, 4$: extracellular loops (N-terminal as E_1); I_i, $i = 1, 2, 3, 4$: intracellular loops (C-terminal as I_4); H_i, $i = 1, 2, 3, 4, 5, 6, 7$: transmembrane helices, and Full: full GPCR sequence. Below are the results of the motif search using the MEME system. A motif was deemed significant if *E-value* < 0.1. This cutoff was chosen to reduce the number of false positive whiles maintaining many hits. Since GPCRs are classified into families based on their sequence and function [Basith et al., 2018], we expected that only a few of the GPCRs under study would share conserved motifs, and that was the case in our analysis (19 significant motifs, Table 6 and Table 11 in Appendix A). We found motifs that were significant across three IUPHAR families by the ligands. These motifs were of length as short as 5 amino acids and as long as 20 amino acids (Table 6 and Table 11). Nevertheless, there were 13 significant motifs that were found across two IUPHAR families by the ligands. These are listed in Table 11 in Appendix A.

Table 6: Human conserved motifs of GPCR sequences across three IUPHAR families found by the MEME system.

Ligands InChI Key	Sequence Region	Length of Region	GPCR UniProt ID	IUPHAR Family	Motif Starting Position in Full Sequence	Motif	E-value
CLQV	Full	371	Q6W5P4	A	196	**GCWARWPDDGYW**	4.40E-02
		463	P43220	B	295		
		912	Q14833	C	538		
FQUA	Full	380	P41145	A	65	**VLSLIFLVGILGNVLVIVVI**	3.90E-38
		400	P35372	A	76		
		373	Q99788	A	46		
		471	P28223	A	80		
		458	P28335	A	59		
		412	P29274	A	12		
		466	P08172	A	29		
		462	P18825	A	57		
		443	P14416	A	40		
		494	Q8IZ08	A	109		
		413	P07550	A	39		
		520	P35368	A	51		
		453	O43194	A	36		
		374	Q8TDU9	A	45		
		407	P25103	A	38		
		418	P37288	A	57		
		487	P35367	A	33		
		442	P24530	A	107		
		361	Q5NUL3	A	46		
		332	P32245	A	50		
		317	Q9NYV8	T2R	12		
		318	P59551	T2R	23		
		466	P48546	B	298		
		460	P11229	A	31		
		346	O14843	A	20		
		312	Q9NYW1	T2R	12		
		299	Q9NYW5	T2R	12		
		307	P59536	T2R	12		
		477	P47871	B	306		
		390	P28222	A	55		
		330	O15552	A	128		
		291	Q9NYV7	T2R	136		
	H_1	23	Q99788	A	46	**VYSLIFVVGILGNVLV**	3.00E-32
		25	P35372	A	76		
		21	O43194	A	36		
		21	Q9NYV8	T2R	12		
		20	O14843	A	20		
		23	P08172	A	29		
		21	O15552	A	13		
		24	P37288	A	57		
		25	P35368	A	51		
		23	P35367	A	33		
		21	Q5NUL3	A	46		
		24	P07550	A	39		
		25	P29274	A	12		
		26	P28335	A	59		
		25	P24530	A	107		
		23	P14416	A	40		
		23	P11229	A	31		
		24	P28223	A	80		
		25	P18825	A	57		
		23	P25103	A	38		
		23	O14842	A	11		
		26	P32245	A	50		
		23	Q9NYW1	T2R	12		
		26	P28222	A	55		

Ligands InChI Key	Sequence Region	Length of Region	GPCR UniProt ID	IUPHAR Family	Motif Starting Position in Full Sequence	Motif	E-value
		21	Q9NYW5	T2R	12		
		21	P59536	T2R	12		
		25	P47871	B	137		
		21	Q8TDU9	A	45		
		21	Q9NYV7	T2R	5		
		28	P41145	A	65		
		21	P08172	A	62		
		25	P32245	A	83		
		24	P07550	A	72		
		22	P28223	A	113		
		24	P41145	A	98		
		21	P11229	A	64		
		21	P35367	A	66		
		22	P35368	A	84		
		20	O43194	A	72		
		26	P28222	A	88		
		26	P18825	A	90		
		24	P29274	A	45		
		23	O14842	A	45		
		25	P35372	A	109		
	H_2	21	Q8TDU9	A	80	**JVSLALADLLVA**	5.90E-47
		21	Q5NUL3	A	78		
		26	P24530	A	140		
		21	Q9NYV7	T2R	49		
		21	P59536	T2R	48		
		22	P37288	A	90		
		21	O15552	A	48		
		21	O14843	A	54		
		22	P25103	A	71		
		25	P47871	B	177		
		20	P48546	B	173		
		21	Q9NYW5	T2R	48		
		22	Q99788	A	78		
		21	P28335	A	92		
		23	P14416	A	73		
		559	P41594	C	176		
		574	Q13255	C	189		
		549	Q14416	C	169		
		555	Q14833	C	183		
MLQF	E_1	550	O00222	C	180	**SPDLSDK**	2.00E-02
		556	Q14831	C	183		
		162	Q03431	B	61		
		554	Q14832	C	175		
		52	Q6W5P4	A	43		
		25	P14416	A	105		
		26	P35462	A	101		
YKMS	E_2*	28	P21917	A	106	**WKFSRAVCD**	2.80E-04
		36	P43220	B	208		
		21	Q14416	C	630		

*We start the region 5 amino acids before and ended 5 amino acids after E_2

4.4 STRUCTURAL COMPARISON

The presence of conserved motifs across the different IUPHAR families led to our decision to compare the 3D structures of the human GPCRs that bind to the same ligand. This was to ascertain whether there are 3D structural similarities either by regions of the GPCRs or the entire

GPCRs. We observed from the 3D structural comparison that comparisons are often done on the entire GPCRs structures if the GPCRs involved are from the same IUPHAR family, whereas regional comparisons are often done when the GPCRs involved are from different IUPHAR families. Table 7 shows the results of the 3D structural comparisons. We performed two types of comparisons:

(i) FATCAT-flexible (Flex in Table 7), which introduces 'twists' between different parts of the proteins which are superimposed independently, and

(ii) FATCAT-rigid (Rigid in Table 7), which uses a rigid-body superposition to align the two structures).

Here FATCAT refers to Flexible structure AlignmenT by Chaining Aligned fragment pairs allowing Twists [Ye & Godzik, 2003; Ye & Godzik, 2004].

In the column "GPCR UniProt ID and IUPHAR Class" of Table 7, for example P16473.A, the characters before the "." represents the Uniprot ID of the GPCR, whereas the character after the "." represents the IUPHAR family. Similarly, in the column "PDB ID and Chain ID," for example 3G04.C, the characters before the "." represent the PDB ID of the GPCR, whereas the character after the "." represents the chain ID. The chain IDs were chosen based on the length of the sequence, i.e., the chain that have the longest sequence was chosen for the comparison.

We observed that over 75% of the pairs compared in Table 7 have both flexible and rigid sequence similarities below 30%, implying they are unlikely to have similar structures. Even for the pair P35462.A and P14416.A with flexible and rigid sequence similarities as high as 88% and 64%, respectively, their rigid RMSD is 9.63, again indicating that there are differences in their structures. The results in Table 7 suggest that two GPCRs, even when they are quite dissimilar in their sequences and structures, can bind to the same ligand. Some of the GPCRs that these ligands bind to have no 3D structures on PDB, and as a result those GPCRs were excluded from further analysis as in the case of AJLF, IKSH, and NKOP (Table 7).

Table 7: Pairwise structural comparison of human GPCR using FATCAT

Ligand	GPCR UniProt ID and IUPHAR Class	PDB ID and Chain ID	RMSD (Å)		Sequence Similarity (%)	
			Flex	Rigid	Flex	Rigid
AJLF	P16473.A, P43220.B	3G04.C, 7LCK.R	2.92	4.87	19	17
DTZD	P43220.B, Q14831.C	7LCK.R, 3MQ4.A	3.63	10.53	20	16
	P43220.B, P21728.A	7LCK.R, 7LJC.R	3.19	3.19	33	33
	Q14831.C, P21728.A	3MQ4.A, 7LJC.R	3.88	10.72	19	15
CLQV	P21453.A, P43220.B	3V2W.A, 7LCK.R	2.84	4.94	27	21
	P21453.A, Q14833.C	3V2W.A, 7E9H.A	3.99	4.51	24	25
	Q14833.C, P43220.B	7E9H.A, 7LCK.R	4.37	6.44	24	24
IKSH	P16473.A, Q03431.B	3G04.C, 6FJ3.A	2.98	4.66	18	22
NKOP	P43220.B, Q13255.C	7LCK.R, 3KS9.A	4.47	8.08	18	17
USZP	P16473.A, P43220.B	3G04.C, 7LCK.R	2.92	4.87	19	17
	P16473.A, P41594.C	3G04.C, 6N52.A	3.06	6.23	15	18
	P43220.B, P41594.C	7LCK.R, 6N52.A	3.83	9.67	25	20
XLWJ	P16473.A, P43220.B	3G04.C, 7LCK.R	2.92	4.87	19	17
	P16473.A, Q03431.B	3G04.C, 6FJ3.A	2.98	4.66	18	22
	P16473.A, P21728.A	3G04.C, 7LJC.R	3.81	3.81	23	23
	P43220.B, Q03431.B	7LCK.R, 6FJ3.A	2.53	3.85	45	50
	P43220.B, P21728.A	7LCK.R, 7LJC.R	3.19	3.19	33	33
	Q03431.B, P21728.A	6FJ3.A, 7LJC.R	3.86	3.03	27	30
	O75899.C, P16473.A	6W2X.B, 3G04.C	3.16	6.04	16	22
	O75899.C, P43220.B	6W2X.B, 7LCK.R	2.92	5.69	24	22
	O75899.C, Q03431.B	6W2X.B, 6FJ3.A	3.20	4.60	24	22
	O75899.C, P21728.A	6W2X.B, 7LJC.R	3.03	3.15	25	26
YKMS	P35462.A, Q14416.C	3PBL.A, 5KZN.A	5.38	11.06	17	17
	P35462.A, P21917.A	3PBL.A, 5WIV.A	1.96	3.95	52	53
	P35462.A, P14416.A	**3PBL.A, 6CM4.A**	**2.15**	**9.63**	**88**	**64**
	P35462.A, P43220.B	3PBL.A, 7LCK.R	6.28	4.44	17	21
	Q14416.C, P21917.A	5KZN.A, 5WIV.A	5.42	6.76	19	19
	Q14416.C, P14416.A	5KZN.A, 6CM4.A	5.23	14.95	17	25
	Q14416.C, P43220.B	5KZN.A, 7LCK.R	5.64	9.41	15	23
	P21917.A, P14416.A	5WIV.A, 6CM4.A	2.72	8.72	52	50
	P21917.A, P43220.B	5WIV.A, 7LCK.R	3.01	4.88	21	23
	P14416.A, P43220.B	6CM4.A, 7LCK.R	3.49	4.44	32	31

4.5 BINDING POCKET COMPARISON AND GPCR LIGAND DOCKING RELATIONSHIP

Since there was no known complex of the 8 ligands bound to the GPCRs on PDB, we had to perform pocket comparison of all possible pairs of pockets; we decided to analyze the relationship between the pocket comparison similarity score, PS-score, and the absolute difference of the binding affinities when the ligand is docked into the pocket. Despite not having a ligand

bound to the GPCR complex on PDB, we observed that, on the average, PS-score over all pairs of pockets of GPCRs that bind the same ligand was 0.287 and above. This value is comparable to the PS-score of 0.284 obtained for the two retinal-binding sites in rhodopsin as mentioned earlier in the Chapter 1. Moreover, among the maximum PS-score values in Table 8, the lowest was 0.349, suggesting that the GPCRs binding to the same ligand would contain similar binding pockets with PS-score no less than 0.349.

We also observed a significant negative correlation between the PS-score and the absolute difference of the binding affinities of the pockets in comparison (Table 9) for most of the ligands. These results suggest that similar binding pockets share similar binding affinity, thus increasing PS-score is associated with similar binding affinity. By extension, the binding pocket residues could be chemically similar.

To verify these results, we used the small control dataset (Table 5) with known interactions and performed binding pocket predictions. The control dataset also showed a significant negative correlation ($r = -0.4547$, p-value = 0.01718) between the PS-score and the absolute difference of the binding affinities of the pairs of pockets across the ligands with a minimum PS-score of 0.114 (Table 8).

Table 8: Summary statistics for PS-scores for all possible pairs of pockets for each ligand.

Ligand	Min	Mean	STD	Max
Control Data	**0.114**	0.487	0.206	0.901
AJLF	0.219	**0.287**	0.0374	0.372
CLQV	0.212	0.326	0.0554	0.528
DTZD	0.218	0.315	0.0498	0.466
IKSH	0.228	0.291	0.0309	**0.349**
NKOP	0.221	0.295	0.0375	0.369
USZP	0.210	0.295	0.0557	0.500
XLWJ	0.218	0.318	0.0570	0.508
YKMS	0.209	0.334	0.0802	0.733

Table 9. Correlation between PS-score and absolute difference of the binding affinities.

Ligand	Correlation	*p*-Value
XLWJ	–0.2776	1.29×10^{-7}
AJLF	–0.7213	4.73×10^{-5}
NKOP	–0.4732	6.23×10^{-3}
IKSH	–0.3501	3.92×10^{-2}
DTZD	–0.1947	4.55×10^{-2}
USZP	–0.2056	7.68×10^{-2}
CLQV	0.1162	0.238
YKMS	0.0318	0.609

4.6 LIGAND BINDING POSE AND CONFORMATION

Figure 9 shows a plot of the $RMSD_{Actual}$ and $RMSD_{Docked}$ of the aligned 3D structures of the ligands as found in complex with the proteins deposited on PDB and when docked into the same binding site (see Section 3.7). We observe from Figure 9 that the two plots exhibit a similar pattern for the control dataset. Additionally, a paired sampled *t*-test on the $RMSD_{Actual}$ and $RMSD_{Docked}$ revealed no significant difference in the mean $RMSD_{Actual}$ (1.2414 ± 0.7654) and $RMSD_{Docked}$ (1.3900 ± 0.5060) (t = –1.5849, df = 26, *p*-value = 0.1251) for the control dataset. These results are good indications for the reliability of the docking results.

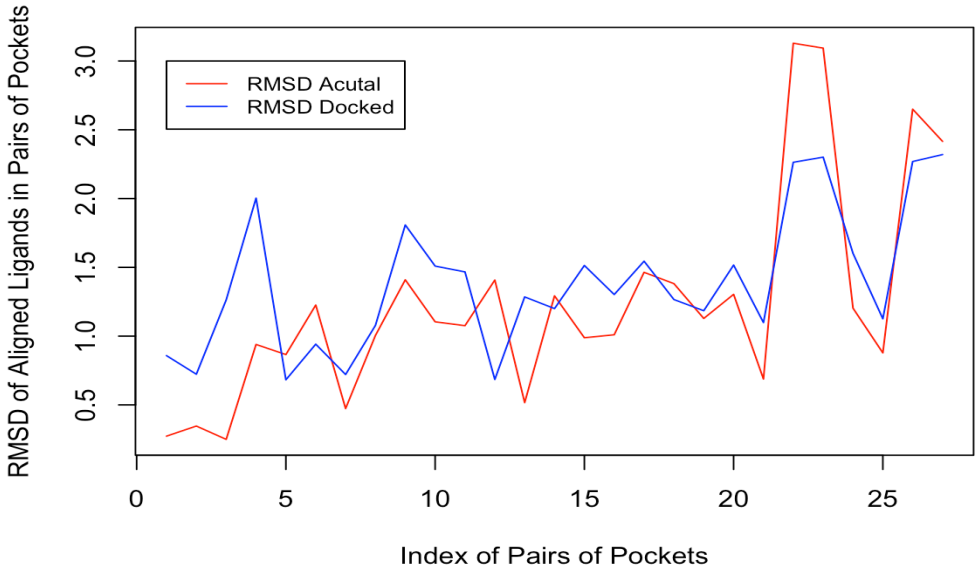


Figure 9: A plot of the $RMSD_{Actual}$ and $RMSD_{Docked}$ pairs of the pockets.

To study the ligands' poses and conformations when binding to their target GPCRs, we chose the pocket pairs with the highest PS-scores, which are all greater than 0.28 (reference PS-score of the two rhodopsin-retinal binding pockets). Generally, we observed varying poses and conformation across the pairs of pockets. A few representative examples were selected to demonstrate the differences in pose and in conformation after docking. The ligand AJLF binds with different poses to pocket 1 of 3G04 and pocket 5 of 7LCK (Figure 10A, B). However, they share very similar conformation (Figure 10C). Similar observations were made for the ligand CLQV (Figure 10D–G). Nevertheless, in the case of the ligand DTZD, there were noticeable differences in the bound conformations of the ligand for the three pockets (Figure 10K). Like in the case of AJLF and CLQV, DTZD binds to pocket 2 of 3MQ4, pocket 4 of 7LCK, and pocket 2 of 7LJC with different poses (Figure 10H–J). However, the Pearson correlation analysis reveals a significant negative correlation (r = –0.0737, *p*-value = 0.02037) between the PS-score and the RMSD of the aligned docked ligands (see Section 3.7) across all pairs of pockets. Similar results were obtained for the control dataset, showing a significant negative correlation (r = –0.5437, *p*-value = 0.0034) between the PS-score and the $RMSD_{Actual}$ across all pairs of pockets. These results

indicate that the more similar the pockets are, the more likely the ligand conformation will be the same.

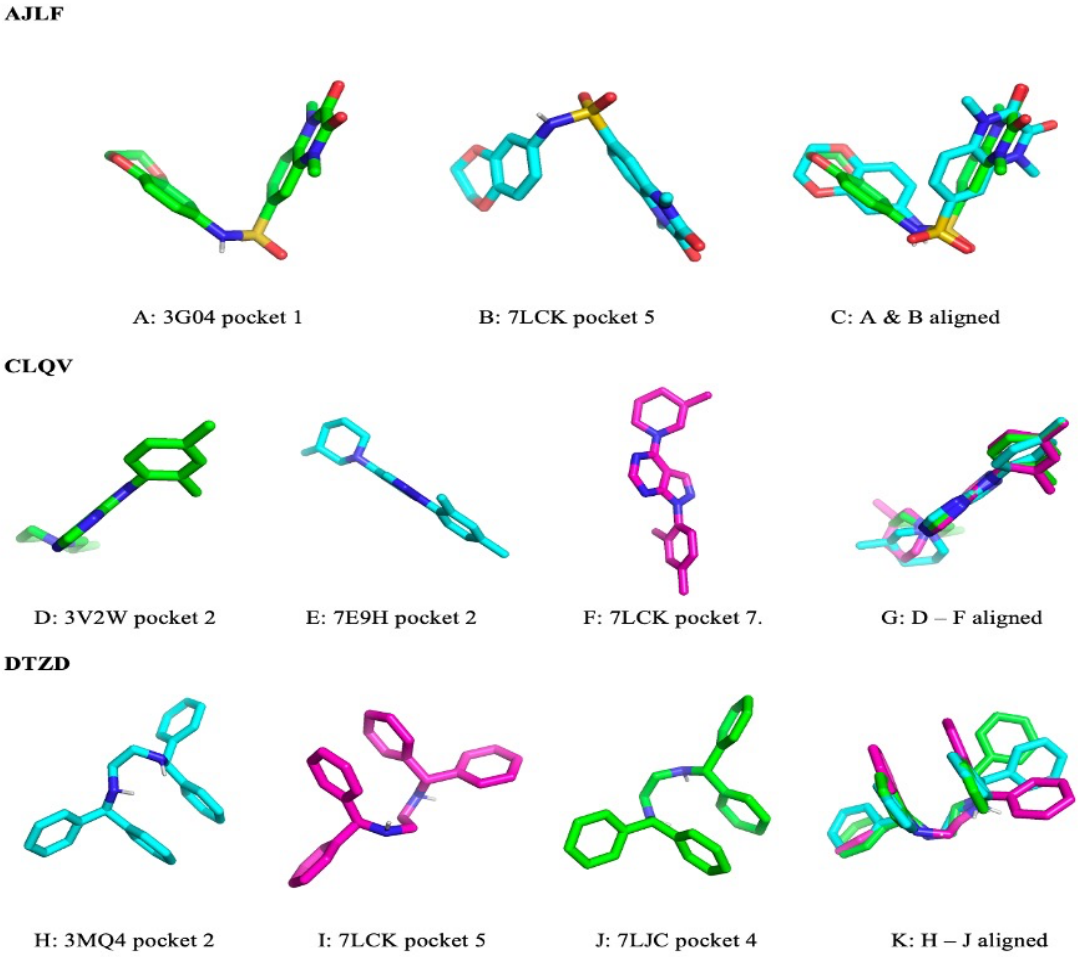
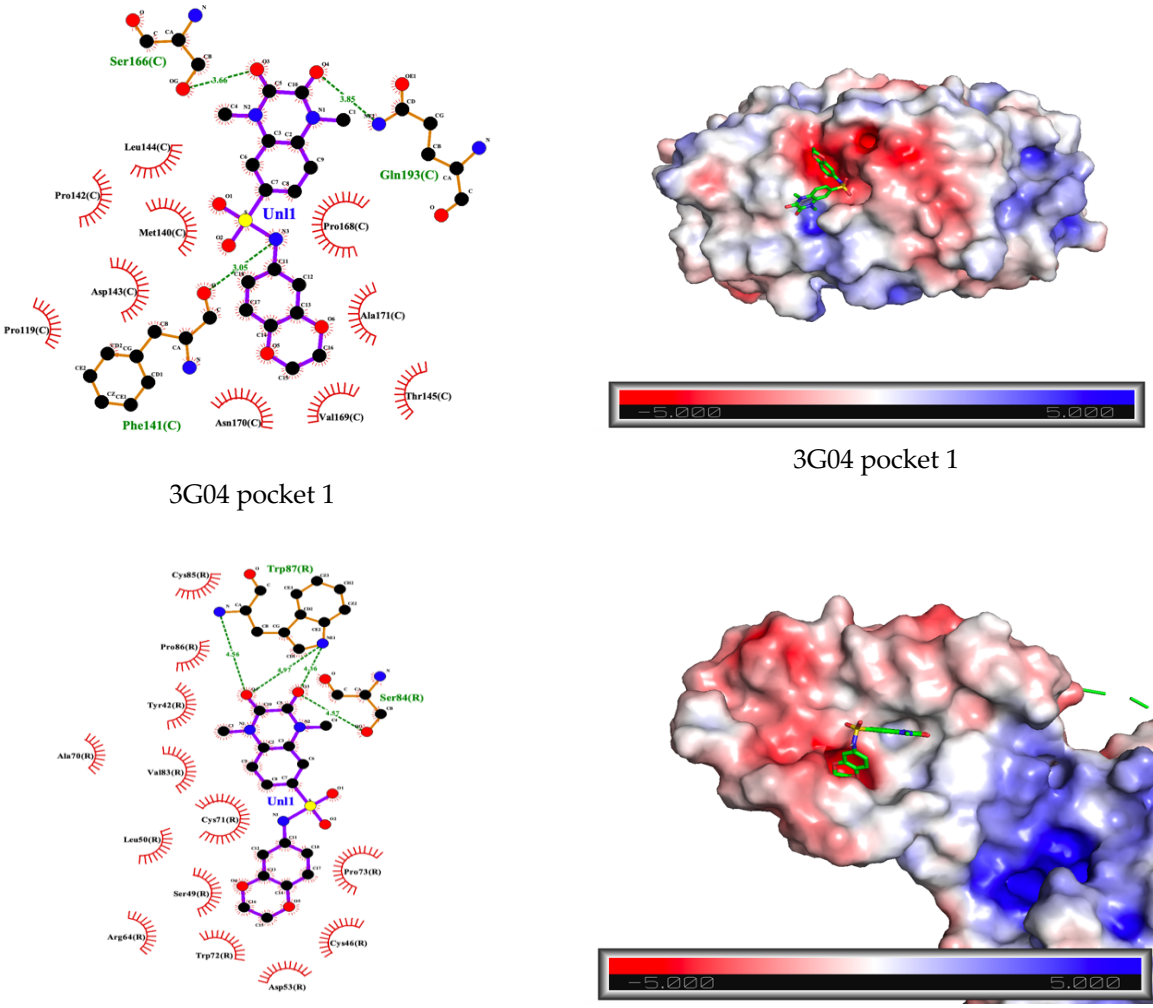


Figure 10: Ligand binding poses and conformations across the different GPCRs binding pockets.

4.7 PROTEIN LIGAND INTERACTION AND POCKET ELECTROSTATIC PROPERTIES

We observed that the ligands tend to bind to similar pockets that share similar residues. The Pearson analysis reveals a significant positive correlation (r = 0.0649, *p*-value = 0.04135) between the PS-score and the number of same residues across all pairs of pockets that interact with the ligands. This suggests that the more similar the pockets are, the more likely a ligand binding to the pockets will interact with the same residues across the pockets. For example, the ligand

AJLF interacts with serine in pockets 1 and 5 of the GPCRs 3G04 and 7LCK, respectively, (Figure 11A).



(**A**) (**B**)

Figure 11: (**A**) AJLF GPCR interaction in the pocket. (**B**) Pocket electrostatic properties with AJLF docked into the pocket.

Generally, we observed that pockets that are similar are also similar in electrostatic properties (Figure 11B), thus increasing PS-score is associated with similar electrostatic properties. The Pearson correlation analysis within the studied ligands revealed mostly negative correlations between the PS-score and the distances obtained from the MS-WHIM scores of the compared pockets (see Section 3.10). Although only half of the correlations were statistically significant

(Table 10), this is an indication that similar pockets may be similar in electrostatic properties and should be further investigated with more ligand pairs in the future.

Table 10: Correlation between PS-scores and Chebyshev distances of MS-WHIM scores for all compared pocket pairs by ligand.

Ligand	Correlation	*p*-Value
YKMS	–0.1936	0.0016
CLQV	–0.2872	0.0029
NKOP	–0.4035	0.0219
USZP	–0.2411	0.0372
AJLF	–0.1964	0.3466
XLWJ	–0.0314	0.5588
DTZD	–0.0306	0.7552
IKSH	0.0263	0.8805

4.8 PREDICTED POCKET AND 3D STRUCTURAL SIMILARITY COMPARISON OVERLAP

We found that some of the predicted pockets of the GPCRs overlap with portions of the GPCRs that were found to be similar from the pairwise 3D structure comparison (see Sections 3.3 and 3.9).

For example, the GPCRs with PDB IDs 3G04 and 7LCK that binds the ligand USZP (Table 7) had portions of their 3D structure that were found to be similar (Figure 12, red parts) overlap with their predicted pockets (Figure 12, ligand USZP). The ligand USZP is docked into a predicted binding pocket of the GPCR. Red part of the GPCRs is where they are 3D structurally similar. We observed from Figure 12 that the position of the ligand USZP (which is docked into a binding pocket) overlaps with the red parts (the parts of the two GPCRs that was found to be 3D structurally similar).

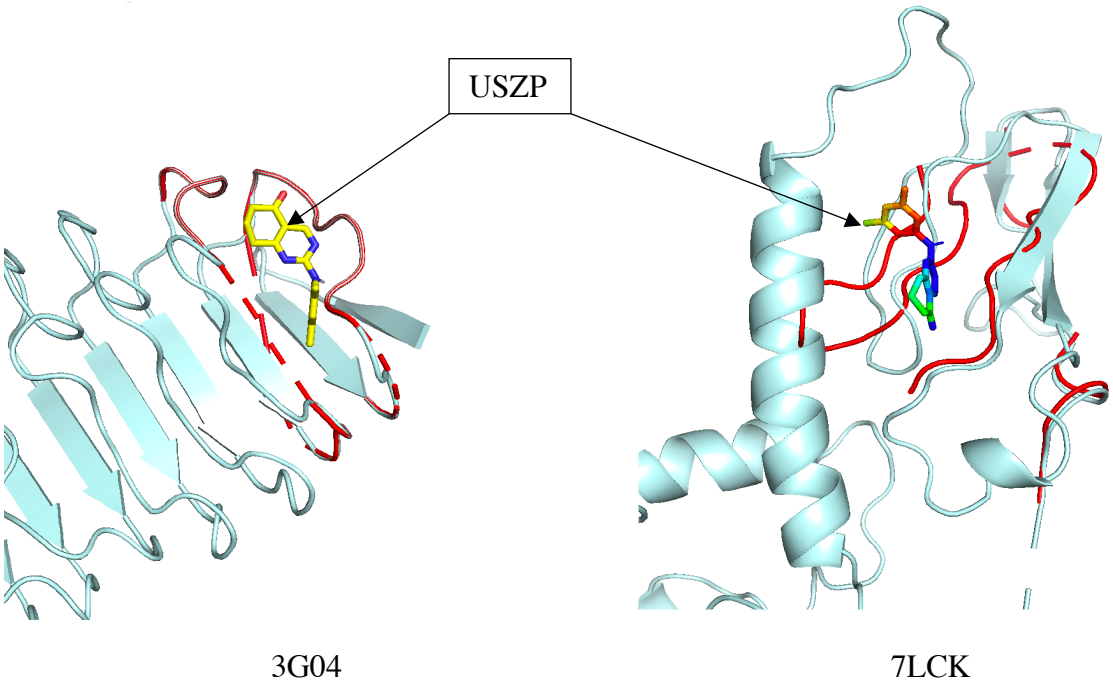


Figure 12: 3D structure of 3G04 vs 7LCK in-bound with the ligand USZP. The ligand USZP is docked to a predicted binding pocket. Red part of the GPCRs is where they are 3D structurally similar.

There is a significant positive correlation ($r = 0.2921$, p-value $= 2.2 \times 10^{-16}$) between the PS-score and the overlap scores across all pairs of pockets. This suggests that the more similar the pockets are, the more likely the pockets are found in a region where the two proteins are structurally similar in their 3D structures.

Chapter 5: Conclusion and Future Work

5.1 Conclusion

We set out to computationally uncover and understand the binding of a single ligand to GPCRs from three different families. As expected, relatively few of such GPCRs of different families share conserved sequence motifs or global structural similarities. However, many share local 3D structural similarities and similar binding pockets. Moreover, the more similar the pockets are, the more likely their binding ligands will interact with the same residues across the pockets, with the same ligand conformation, and similar binding affinities across the pockets. In addition, the more similar the pockets are, the more likely the electrostatic properties of the pockets will be similar, and the more likely the pockets are found in a region where the two proteins are structurally similar in their 3D structures. These findings can be taken advantage of to further develop protein function inference, drug toxicity prediction, and discovery of unwanted cross reactivity to speed up the process of drug repurposing and new drug development.

5.2 Future research and proposed approaches

While the focus of this dissertation is on ligands that bind to GPCRs from three different IUPHAR families, it has several extensions. We can assess whether the discovered conserved motifs overlap with the predicted binding pockets of the ligands. This will help us determine whether the motifs are involved in the binding of the same ligand to the GPCRs of different IUPHAR families.

Furthermore, we can expand on this work by clustering binding pockets of GPCRs based on pocket electrostatic properties, binding affinities, pocket sequence and structural features, and then ranking the pockets in each cluster based on their pairwise pocket comparison scores to identify the most likely pockets that can bind to the same ligand.

Finally, we should also extend our analysis to include the three ligands, namely BYBL, FQUA, and MLQF, that bind to GPCRs from three different IUPHAR families but were excluded from our analysis to reduce the number of pairwise comparisons. These ligands were estimated to generate over 30,000 binding pocket pairs in total, which would increase the computational time exponentially. To complete our analysis within reasonable time, we had to omit the three ligands from this study. The issue of excessive computational time can be circumvented by parallelizing the work done in the sections of binding pocket prediction and comparison, structural comparison, protein ligand interaction, and predicted pocket and 3D structural similarity comparison overlap, all of which require substantial computing resources. Designing such a parallelization scheme is an interesting computational challenge, but successfully implementing it would allow us to also investigate the 16000+ ligands that bind to GPCRs from two different IUPHAR families. This will, in turn, help us obtain more complete and definitive answers to our central question of what makes GPCRs from different families bind to the same ligand.

References

Alexander, S. P. H., Christopoulos, A., Davenport, A. P., Kelly, E., Mathie, A., Peters, J. A., Veale, E. L., Armstrong, J. F., Faccenda, E., Harding, S. D., Pawson, A. J., Sharman, J. L., Southan, C., Davies, J. A., Abbracchio, M. P., Alexander, W., Al-hosaini, K., Bäck, M., Beaulieu, J. M., … Yao, C. (2019). THE CONCISE GUIDE TO PHARMACOLOGY 2019/20: G protein-coupled receptors. *British Journal of Pharmacology*, *176*(S1), S21–S141. https://doi.org/10.1111/bph.14748

Armstrong, J. F., Faccenda, E., Harding, S. D., Pawson, A. J., Southan, C., Sharman, J. L., Campo, B., Cavanagh, D. R., Alexander, S. P. H., Davenport, A. P., Spedding, M., & Davies, J. A. (2020). The IUPHAR/BPS Guide to PHARMACOLOGY in 2020: Extending immunopharmacology content and introducing the IUPHAR/MMV Guide to MALARIA PHARMACOLOGY. *Nucleic Acids Research*, *48*(D1), D1006–D1021. https://doi.org/10.1093/nar/gkz951

Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., Ren, J., Li, W. W., & Noble, W. S. (2009). MEME Suite: Tools for motif discovery and searching. *Nucleic Acids Research*, *37*(SUPPL. 2), 202–208. https://doi.org/10.1093/nar/gkp335

Bailey, T. L., Johnson, J., Grant, C. E., & Noble, W. S. (2015). The MEME Suite. *Nucleic Acids Research*, *43*(W1), W39–W49. https://doi.org/10.1093/nar/gkv416

Bailey, T. L., Williams, N., Misleh, C., & Li, W. W. (2006). MEME: Discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Research*, *34*(WEB. SERV. ISS.), 369–373. https://doi.org/10.1093/nar/gkl198

Basith, S., Cui, M., Macalino, S. J. Y., Park, J., Clavio, N. A. B., Kang, S., & Choi, S. (2018).

Exploring G protein-coupled receptors (GPCRs) ligand space via cheminformatics approaches: Impact on rational drug design. *Frontiers in Pharmacology*, *9*(MAR), 1–26. https://doi.org/10.3389/fphar.2018.00128

Bateman, A. (2019). UniProt: A worldwide hub of protein knowledge. *Nucleic Acids Research*, *47*(D1), D506–D515. https://doi.org/10.1093/nar/gky1049

Begum, K., Mohl, J. E., Ayivor, F., Perez, E. E., & Leung, M. Y. (2020). GPCR-PEnDB: A database of protein sequences and derived features to facilitate prediction and classification of G protein-coupled receptors. *Database*, *2020*, 1–12. https://doi.org/10.1093/database/baaa087

Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D., & Zardecki, C. (2002). The protein data bank. *Acta Crystallographica Section D: Biological Crystallography*, *58*(6 I), 899–907. https://doi.org/10.1107/S0907444902003451

Binkowski TA, Naghibzadeh S, Liang J (2003) CASTp: computed atlas of surface topography of proteins. Nucleic Acids Res 31(13): 3352–3355.

Bravi, G., Gancia, E., Mascagni, P., Pegna, M., Todeschini, R., & Zaliani, A. (1997). MS-WHIM, new 3D theoretical descriptors derived from molecular surface properties: A comparative 3D QSAR study in a series of steroids. *Journal of Computer-Aided Molecular Design*, *11*(1), 79–92. https://doi.org/10.1023/A:1008079512289

Brylinski, M., & Skolnick, J. (2008). A threading-based method (FINDSITE) for ligand-binding site prediction and functional annotation. *Proceedings of the National Academy of Sciences*

of the United States of America, *105*(1), 129–134. https://doi.org/10.1073/pnas.0707684105

Capra JA, Laskowski RA, Thornton JM, Singh M, Funkhouser TA (2009) Predicting protein ligand binding sites by combining evolutionary sequence conservation and 3D structure. PLoS Comput Biol 5(12):e1000585.

Chan, W. K. B., Zhang, H., Yang, J., Brender, J. R., Hur, J., Ozgur, A., & Zhang, Y. (2015). GLASS: A comprehensive database for experimentally validated GPCR-ligand associations. *Bioinformatics*, *31*(18), 3035–3042. https://doi.org/10.1093/bioinformatics/btv302

Chang DT, Oyang YJ, Lin JH (2005) MEDock: a web server for efficient prediction of ligand binding sites based on a novel optimization algorithm. Nucleic Acids Res 33(Web Server issue): W233–W238.

Christopher, J. A., Orgován, Z., Congreve, M., Doré, A. S., Errey, J. C., Marshall, F. H., ... & Ferenczy, G. G. (2018). Structure-based optimization strategies for G protein-coupled receptor (GPCR) allosteric modulators: a case study from analyses of new metabotropic glutamate receptor 5 (mGlu5) X-ray structures. Journal of medicinal chemistry, 62(1), 207-222.

Ciancetta, A., Sabbadin, D., Federico, S., Spalluto, G., & Moro, S. (2015). Advances in Computational Techniques to Study GPCR-Ligand Recognition. *Trends in Pharmacological Sciences*, *36*(12), 878–890. https://doi.org/10.1016/j.tips.2015.08.006

Clynen, E., Swijsen, A., Raijmakers, M., Hoogland, G., & Rigo, J. M. (2014). Neuropeptides as Targets for the Development of Anticonvulsant Drugs. *Molecular Neurobiology*, *50*(2), 626–646. https://doi.org/10.1007/s12035-014-8669-x

Daniel Osorio. (2021). *Package "Peptides" Title Calculate Indices and Theoretical Physicochemical Properties of Protein Sequences Suggests testthat (>= 2.1.0)*. https://doi.org/10.32614/RJ-2015-001

Dankwah, K. O., Mohl, J. E., Begum, K., & Leung, M. Y. (2021, December). Understanding the binding of the same ligand to GPCRs of different families. In 2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 2494-2501). IEEE.

Dankwah, K. O., Mohl, J. E., Begum, K., & Leung, M.-Y. (2022). What makes gpcrs from different families bind to the same ligand? Biomolecules, 12(7), 863. https://doi.org/10.3390/biom12070863

Diepenhorst, N., Rueda, P., Cook, A. E., Pastoureau, P., Sabatini, M., & Langmead, C. J. (2018). G protein-coupled receptors as anabolic drug targets in osteoporosis. *Pharmacology and Therapeutics*, *184*(November 2017), 1–12. https://doi.org/10.1016/j.pharmthera.2017.10.015

Feng, X., Ambia, J., Chen, K. Y. M., Young, M., & Barth, P. (2017). Computational design of ligand-binding membrane receptors with high selectivity. *Nature Chemical Biology*, *13*(7), 715–723. https://doi.org/10.1038/nchembio.2371

Forli, S., Huey, R., Pique, M. E., Sanner, M. F., Goodsell, D. S., & Olson, A. J. (2016). Computational protein–ligand docking and virtual drug screening with the AutoDock suite. *Nature Protocols*, *11*(5), 905–919. https://doi.org/10.1038/nprot.2016.051

Freudenberg, J. M., Dunham, I., Sanseau, P., & Rajpal, D. K. (2018). Uncovering new disease indications for G-protein coupled receptors and their endogenous ligands. *BMC Bioinformatics*, *19*(1), 1–11. https://doi.org/10.1186/s12859-018-2392-y

Friesner, R. A., Banks, J. L., Murphy, R. B., Halgren, T. A., Klicic, J. J., Mainz, D. T., Repasky, M. P., Knoll, E. H., Shelley, M., Perry, J. K., Shaw, D. E., Francis, P., & Shenkin, P. S. (2004). Glide: A New Approach for Rapid, Accurate Docking and Scoring. 1. Method and Assessment of Docking Accuracy. *Journal of Medicinal Chemistry*, *47*(7), 1739–1749. https://doi.org/10.1021/jm0306430

Friesner, R. A., Murphy, R. B., Repasky, M. P., Frye, L. L., Greenwood, J. R., Halgren, T. A., Sanschagrin, P. C., & Mainz, D. T. (2006). Extra precision glide: Docking and scoring incorporating a model of hydrophobic enclosure for protein-ligand complexes. *Journal of Medicinal Chemistry*, *49*(21), 6177–6196. https://doi.org/10.1021/jm051256o

Fu, D. Y., & Meiler, J. (2018). Rosetta Ligand Ensemble: A Small-Molecule Ensemble-Driven Docking Approach. *ACS Omega*, *3*(4), 3655–3664. https://doi.org/10.1021/acsomega.7b02059

Fu, Y., Zhao, J., & Chen, Z. (2018). Insights into the Molecular Mechanisms of Protein-Ligand Interactions by Molecular Docking and Molecular Dynamics Simulation: A Case of Oligopeptide Binding Protein. *Computational and Mathematical Methods in Medicine*, *2018*. https://doi.org/10.1155/2018/3502514

Gad, A. A., & Balenga, N. (2020). The Emerging Role of Adhesion GPCRs in Cancer. *ACS Pharmacology and Translational Science*, *3*(1), 29–42. https://doi.org/10.1021/acsptsci.9b00093

Gao, M., & Skolnick, J. (2013). APoc: Large-scale identification of similar protein pockets. *Bioinformatics*, *29*(5), 597–604. https://doi.org/10.1093/bioinformatics/btt024

Ghersi D, Sanchez R (2009) EasyMIFS and SiteHound: a toolkit for the identification of ligand-

binding sites in protein structures. Bioinformatics 25(23):3185–3186.

Gilson, M. K., Liu, T., Baitaluk, M., Nicola, G., Hwang, L., & Chong, J. (2016). BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic Acids Research*, *44*(D1), D1045–D1053. https://doi.org/10.1093/nar/gkv1072

Goodsell, D. S., Zardecki, C., Di Costanzo, L., Duarte, J. M., Hudson, B. P., Persikova, I., Segura, J., Shao, C., Voigt, M., Westbrook, J. D., Young, J. Y., & Burley, S. K. (2019). RCSB Protein Data Bank: Enabling biomedical research and drug discovery. *Protein Science*, *September 2019*, 52–65. https://doi.org/10.1002/pro.3730

Govindaraj, R. G., & Brylinski, M. (2018). Comparative assessment of strategies to identify similar ligand-binding pockets in proteins. *BMC Bioinformatics*, *19*(1), 1–17. https://doi.org/10.1186/s12859-018-2109-2

Guipponi, M., Chentouf, A., Webling, K. E. B., Freimann, K., Crespel, A., Nobile, C., Lemke, J. R., Hansen, J., Dorn, T., Lesca, G., Ryvlin, P., Hirsch, E., Rudolf, G., Rosenberg, D. S., Weber, Y., Becker, F., Helbig, I., Muhle, H., Salzmann, A., … Antonarakis, S. E. (2014). Galanin pathogenic mutations in temporal lobe epilepsy. *Human Molecular Genetics*, *24*(11), 3082–3091. https://doi.org/10.1093/hmg/ddv060

Gutteridge A, Bartlett GJ, Thornton JM (2003) Using a neural network and spatial clustering to predict the location of active sites in enzymes. J Mol Biol 330(4): 719–734.

Hasegawa, H., & Holm, L. (2009). Advances and pitfalls of protein structural alignment. *Current Opinion in Structural Biology*, *19*(3), 341–348. https://doi.org/10.1016/j.sbi.2009.04.003

Hassan, M. M. (2018). Deep Learning Models For Scoring Protein-Ligand Interaction Energies.

In *Open Access Theses & Dissertations* (Issue 1447). https://digitalcommons.utep.edu/open_etd/1447

Holm, L., & Laakso, L. M. (2016). Dali server update. *Nucleic Acids Research*, *44*(W1), W351–W355. https://doi.org/10.1093/nar/gkw357

Holm, L., & Sander, C. (1993). Protein structure comparison by alignment of distance matrices. In *Journal of Molecular Biology* (Vol. 233, Issue 1, pp. 123–138). https://doi.org/10.1006/jmbi.1993.1489

Huang, B. (2009). Metapocket: A meta approach to improve protein ligand binding site prediction. *OMICS A Journal of Integrative Biology*, *13*(4), 325–330. https://doi.org/10.1089/omi.2009.0045

Huang, B., & Schroeder, M. (2006). LIGSITEcsc: Predicting ligand binding sites using the Connolly surface and degree of conservation. *BMC Structural Biology*, *6*, 1–11. https://doi.org/10.1186/1472-6807-6-19

Huang, Y., Todd, N., & Thathiah, A. (2017). The role of GPCRs in neurodegenerative diseases: avenues for therapeutic intervention. *Current Opinion in Pharmacology*, *32*, 96–110. https://doi.org/10.1016/j.coph.2017.02.001

Insel, P. A., Sriram, K., Wiley, S. Z., Wilderman, A., Katakia, T., McCann, T., Yokouchi, H., Zhang, L., Corriden, R., Liu, D., Feigin, M. E., French, R. P., Lowy, A. M., & Murray, F. (2018). GPCRomics: GPCR expression in cancer cells and tumors identifies new, potential biomarkers and therapeutic targets. *Frontiers in Pharmacology*, *9*(MAY), 1–11. https://doi.org/10.3389/fphar.2018.00431

Jiménez, J., Doerr, S., Martínez-Rosell, G., Rose, A. S., & De Fabritiis, G. (2017). DeepSite:

Protein-binding site predictor using 3D-convolutional neural networks. *Bioinformatics*, *33*(19), 3036–3042. https://doi.org/10.1093/bioinformatics/btx350

Jo, M., & Jung, S. T. (2016). Engineering therapeutic antibodies targeting G-protein-coupled receptors. *Experimental & Molecular Medicine*, *48*(September 2015), e207. https://doi.org/10.1038/emm.2015.105

Jurrus, E., Engel, D., Star, K., Monson, K., Brandi, J., Felberg, L. E., Brookes, D. H., Wilson, L., Chen, J., Liles, K., Chun, M., Li, P., Gohara, D. W., Dolinsky, T., Konecny, R., Koes, D. R., Nielsen, J. E., Head-Gordon, T., Geng, W., … Baker, N. A. (2018). Improvements to the APBS biomolecular solvation software suite. *Protein Science*, *27*(1), 112–128. https://doi.org/10.1002/pro.3280

Kinoshita, K., Furui, J., & Nakamura, H. (2002). Identification of protein functions from a molecular surface database, eF-site. *Journal of Structural and Functional Genomics*, *2*(1), 9–22. https://doi.org/10.1023/A:1011318527094

Kinoshita, K., Murakami, Y., & Nakamura, H. (2007). EF-seek: Prediction of the functional sites of proteins by searching for similar electrostatic potential and molecular surface shape. *Nucleic Acids Research*, *35*(SUPPL.2), 398–402. https://doi.org/10.1093/nar/gkm351

Konc, J., & Janežič, D. (2010). ProBiS algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics*, *26*(9), 1160–1168. https://doi.org/10.1093/bioinformatics/btq100

Kramer, C., Spinn, A., & Liedl, K. R. (2014). Charge anisotropy: Where atomic multipoles matter most. *Journal of Chemical Theory and Computation*, *10*(10), 4488–4496. https://doi.org/10.1021/ct5005565

Krivák, R., & Hoksza, D. (2018). P2Rank: machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure. *Journal of Cheminformatics*, *10*(1), 1–12. https://doi.org/10.1186/s13321-018-0285-8

Kukol, A. (2014). Molecular modeling of proteins: Second edition. *Molecular Modeling of Proteins: Second Edition*, *1215*, 1–474. https://doi.org/10.1007/978-1-4939-1465-4

Lappano, R., & Maggiolini, M. (2012). GPCRs and cancer. *Acta Pharmacologica Sinica*, *33*(3), 351–362. https://doi.org/10.1038/aps.2011.183

Laurie AT, Jackson RM (2005) Q-SiteFinder: an energy-based method for the prediction of protein–ligand bind.

Laskowski, R. A., & Swindells, M. B. (2011). LigPlot+: Multiple ligand-protein interaction diagrams for drug discovery. *Journal of Chemical Information and Modeling*, *51*(10), 2778–2786. https://doi.org/10.1021/ci200227u

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

Lee, D., Redfern, O., & Orengo, C. (2007). Predicting protein function from sequence and structure. *Nature Reviews Molecular Cell Biology*, *8*(12), 995–1005. https://doi.org/10.1038/nrm2281

Lee, H. S., & Im, W. (2016). G-LoSA: An efficient computational tool for local structure-centric biological studies and drug design. *Protein Science*, *25*(4), 865–876. https://doi.org/10.1002/pro.2890

Lee, H. S., & Im, W. (2017). G-LoSA for prediction of protein-ligand binding sites and structures. *Methods in Molecular Biology*, *1611*, 97–108. https://doi.org/10.1007/978-1-

4939-7015-5_8

Li J, Fu A, Zhang L (2019) An overview of scoring functions used for protein–ligand interactions in molecular docking, Interdisciplinary Sciences: Computational Life Sciences 1–9.

López, G., Valencia, A., & Tress, M. L. (2007). Firestar-prediction of functionally important residues using structural templates and alignment reliability. *Nucleic Acids Research*, *35*(SUPPL.2), 573–577. https://doi.org/10.1093/nar/gkm297

Malhotra, S., & Karanicolas, J. (2017). When Does Chemical Elaboration Induce a Ligand To Change Its Binding Mode? *Journal of Medicinal Chemistry*, *60*(1), 128–145. https://doi.org/10.1021/acs.jmedchem.6b00725

Matera, M. G., Page, C., & Rinaldi, B. (2018). β2-Adrenoceptor signalling bias in asthma and COPD and the potential impact on the comorbidities associated with these diseases. *Current Opinion in Pharmacology*, *40*(May), 142–146. https://doi.org/10.1016/j.coph.2018.04.012

Mazarati, A., Langel, Ü., & Bartfai, T. (2001). Galanin: An endogenous anticonvulsant? *Neuroscientist*, *7*(6), 506–517. https://doi.org/10.1177/107385840100700607

Mehio W, Kemp GJ, Taylor P, Walkinshaw MD (2010) Identification of protein binding surfaces using surface triplet propensities. Bioinformatics 26(20):2549–2555.

Morita M, Nakamura S, Shimizu K (2008) Highly accurate method for ligand-binding site prediction in unbound state (apo) protein structures. Proteins 73(2):468–479

Morris, G. M., Ruth, H., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., & Olson, A. J. (2009). AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *Journal of Computational Chemistry*, *30*(16), 2785–2791.

https://doi.org/10.1002/jcc.21256

Nagarathnam, B., Kannan, S., Dharnidharka, V., Balakrishnan, V., Archunan, G., & Sowdhamini, R. (2011). Insights from the analysis of conserved motifs and permitted amino acid exchanges in the human, the fly and the worm GPCR clusters. *Bioinformation*, *7*(1), 15–20. https://doi.org/10.6026/97320630007015

Nayal M, Honig B (2006) On the nature of cavities on protein surfaces: application to the identification of drug-binding sites. Proteins 63(4):892–906.

Ngan CH, Hall DR, Zerbe B, Grove LE, Kozakov D, Vajda S (2012) FTSite: high accu- racy detection of ligand binding sites on unbound protein structures. Bioinformatics 28(2):286–287.

Nguyen, D. D., Xiao, T., Wang, M., & Wei, G. W. (2017). Rigidity Strengthening: A Mechanism for Protein-Ligand Binding. *Journal of Chemical Information and Modeling*, *57*(7), 1715–1721. https://doi.org/10.1021/acs.jcim.7b00226

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., & Hutchison, G. R. (2011). Open Babel. *Journal of Cheminformatics*, *3*(33), 1–14. https://jcheminf.biomedcentral.com/track/pdf/10.1186/1758-2946-3-33

Prlić, A., Bliven, S., Rose, P. W., Bluhm, W. F., Bizon, C., Godzik, A., & Bourne, P. E. (2010). Pre-calculated protein structure alignments at the RCSB PDB website. *Bioinformatics*, *26*(23), 2983–2985. https://doi.org/10.1093/bioinformatics/btq572

Raschka, S., & Kaufman, B. (2020). Machine learning and AI-based approaches for bioactive ligand discovery and GPCR-ligand recognition. *Methods*, *180*(January), 89–110. https://doi.org/10.1016/j.ymeth.2020.06.016

Roche, Daniel B., Buenavista, M. T., Tetchner, S. J., & McGuffin, L. J. (2011). The IntFOLD server: An integrated web resource for protein fold recognition, 3D model quality assessment, intrinsic disorder prediction, domain prediction and ligand binding site prediction. *Nucleic Acids Research*, *39*(SUPPL. 2), 171–176. https://doi.org/10.1093/nar/gkr184

Roche DB, Buenavista MT, Mcguffi n LJ (2012) FunFOLDQA: a quality assessment tool for protein- ligand binding site residue predictions. PLoS One 7: e38219

Roche DB, Buenavista MT, Mcguffi n LJ (2013) The FunFOLD2 server for the prediction of protein- ligand interactions. Nucleic Acids Res 41: W303–W307

Roche, Daniel Barry, & McGuffin, L. J. (2016). In silico identifi cation and characterization of protein-ligand binding sites. *Methods in Molecular Biology*, *1414*, 1–21. https://doi.org/10.1007/978-1-4939-3569-7_1

Römpler, H., Yu, H. T., Arnold, A., Orth, A., & Schöneberg, T. (2006). Functional consequences of naturally occurring DRY motif variants in the mammalian chemoattractant receptor GPR33. *Genomics*, *87*(6), 724–732. https://doi.org/10.1016/j.ygeno.2006.02.009

Ruiz-Carmona, S., Alvarez-Garcia, D., Foloppe, N., Garmendia-Doval, A. B., Juhos, S., Schmidtke, P., Barril, X., Hubbard, R. E., & Morley, S. D. (2014). rDock: A Fast, Versatile and Open Source Program for Docking Ligands to Proteins and Nucleic Acids. *PLoS Computational Biology*, *10*(4), 1–7. https://doi.org/10.1371/journal.pcbi.1003571

Sael, L., & Kihara, D. (2012). Detecting Local Ligand-Binding Site Similarity in Non-Homologous Proteins by Surface Patch Comparison. *Structure, Function, and Bioinformatics, 80*(4), 11177–11195. https://doi.org/10.1002/prot.24018

Schiöth, H. B., & Fredriksson, R. (2005). The GRAFS classification system of G-protein coupled receptors in comparative perspective. *General and Comparative Endocrinology*, *142*(1-2 SPEC. ISS.), 94–101. https://doi.org/10.1016/j.ygcen.2004.12.018

Schmitt, S., Kuhn, D., & Klebe, G. (2002). A new method to detect related function among proteins independent of sequence and fold homology. *Journal of Molecular Biology*, *323*(2), 387–406. https://doi.org/10.1016/S0022-2836(02)00811-2

Sehnal D, Rose A.S., Kovca J., Burley S.K., Velankar S. (2018) Mol*: Towards a common library and tools for web molecular graphics MolVA/EuroVis Proceedings. doi:10.2312/molva.20181103)

Seo, S., Choi, J., Ahn, S. K., Kim, K. W., Kim, J., Choi, J., Kim, J., & Ahn, J. (2018a). Prediction of GPCR-Ligand Binding Using Machine Learning Algorithms. *Computational and Mathematical Methods in Medicine*, *2018*. https://doi.org/10.1155/2018/6565241

Seo, S., Choi, J., Ahn, S. K., Kim, K. W., Kim, J., Choi, J., Kim, J., & Ahn, J. (2018b). Prediction of GPCR-Ligand Binding Using Machine Learning Algorithms. *Computational and Mathematical Methods in Medicine*, *2018*. https://doi.org/10.1155/2018/6565241

Shatsky M., Nussinov R., Wolfson H.J. (2002) MultiProt — A Multiple Protein Structural Alignment Algorithm. In: Guigó R., Gusfield D. (eds) Algorithms in Bioinformatics. WABI 2002. Lecture Notes in Computer Science, vol 2452. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45784-4_18

Shindyalov, I. N., & Bourne, P. E. (1998). Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, *11*(9), 739–747. https://doi.org/10.1093/protein/11.9.739

Shulman-Peleg, A., Nussinov, R., & Wolfson, H. J. (2005). SiteEngines: Recognition and comparison of binding sites and protein-protein interfaces. *Nucleic Acids Research*, *33*(SUPPL. 2), 337–341. https://doi.org/10.1093/nar/gki482

Singh, G., Inoue, A., Gutkind, J. S., Russell, R. B., & Raimondi, F. (2019). PRECOG: PREdicting COupling probabilities of G-protein coupled receptors. *Nucleic Acids Research*, *47*(W1), W395–W401. https://doi.org/10.1093/nar/gkz392

Sinha, N., & Smith-Gill, S. J. (2002). Electrostatics in protein binding and function. Current Protein and Peptide Science, 3(6), 601-614.

Skolnick, J., & Brylinski, M. (2009). FINDSITE: A combined evolution/structure-based approach to protein function prediction. *Briefings in Bioinformatics*, *10*(4), 378–391. https://doi.org/10.1093/bib/bbp017

Soga S, Shirai H, Kobori M, Hirayama N (2007) Use of amino acid composition to pre- dict ligand-binding sites. J Chem Inf Model 47(2):400–406.

Spitzer, R., & Jain, A. N. (2012). Surflex-Dock: Docking benchmarks and real-world application. *Journal of Computer-Aided Molecular Design*, *26*(6), 687–699. https://doi.org/10.1007/s10822-011-9533-y

Srinivasan, S., Guixà-González, R., Cordomí, A., & Garriga, P. (2019). Ligand Binding Mechanisms in Human Cone Visual Pigments. *Trends in Biochemical Sciences*, *44*(7), 629–639. https://doi.org/10.1016/j.tibs.2019.02.001

Taylor, R. D., Jewsbury, P. J., & Essex, J. W. (2002). A review of protein-small molecule docking methods - Taylor-JCAMD2002.pdf. *Journal of Computer-Aided Molecular Design*, *16*, 151–166.

https://link.springer.com/content/pdf/10.1023%2FA%3A1020155510718.pdf%0Ahttp://infochimie.u-strasbg.fr/master/tutochemo/TP10/Docking_PDF_2010/Bibliography/Taylor-JCAMD2002.pdf

Teilum, K., Olsen, J. G., & Kragelund, B. B. (2009). Functional aspects of protein flexibility. *Cellular and Molecular Life Sciences*, *66*(14), 2231–2247. https://doi.org/10.1007/s00018-009-0014-6

Tripathi A, Kellogg GE (2010) A novel and efficient tool for locating and characterizing protein cavities and binding sites. Proteins. Mar;78(4):825-42. doi: 10.1002/prot.22608. PMID: 19847777; PMCID: PMC2811767.

Trott, O., & Olson, A. J. (2009). AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry*, *31*(2), NA-NA. https://doi.org/10.1002/jcc.21334

Vascon, F., Gasparotto, M., Giacomello, M., Cendron, L., Bergantino, E., Filippini, F., & Righetto, I. (2020). Protein electrostatics: From computational and structural analysis to discovery of functional fingerprints and biotechnological design. *Computational and Structural Biotechnology Journal*, *18*, 1774–1789. https://doi.org/10.1016/j.csbj.2020.06.029

Wass, M. N., Kelley, L. A., & Sternberg, M. J. E. (2010). 3DLigandSite: Predicting ligand-binding sites using similar structures. *Nucleic Acids Research*, *38*(SUPPL. 2), 469–473. https://doi.org/10.1093/nar/gkq406

Wass, M. N., & Sternberg, M. J. E. (2009). Prediction of ligand binding sites using homologous structures and conservation at CASP8. *Proteins: Structure, Function and Bioinformatics*,

77(SUPPL. 9), 147–151. https://doi.org/10.1002/prot.22513

Weisel M, Proschak E, Schneider G (2007) PocketPicker: analysis of ligand binding-sites with shape descriptors. Chem Cent J. Mar 13;1:7. doi: 10.1186/1752-153X-1-7. PMID: 17880740; PMCID: PMC1994066.

Xie, L., & Bourne, P. E. (2008). Detecting evolutionary relationships across existing fold space, using sequence order-independent profile-profile alignments. *Proceedings of the National Academy of Sciences of the United States of America*, *105*(14), 5441–5446. https://doi.org/10.1073/pnas.0704422105

Xie ZR, Hwang MJ (2012) Ligand-binding site prediction using ligand-interacting and binding site-enriched protein triangles. Bioinformatics 28(12):1579–1585.

Xie ZR, Liu CK, Hsiao FC, Yao A, Hwang MJ (2013) LISE: a server using ligand-interacting and site-enriched protein triangles for predic- tion of ligand-binding sites. Nucleic Acids Res 41(Web Server issue):W292–W296

Ye, Y., & Godzik, A. (2003). Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics*, *19*(SUPPL. 2). https://doi.org/10.1093/bioinformatics/btg1086

Ye, Y., & Godzik, A. (2004). FATCAT: A web server for flexible structure comparison and structure similarity searching. *Nucleic Acids Research*, *32*(WEB SERVER ISS.), 582–585. https://doi.org/10.1093/nar/gkh430

Yeturu, K., & Chandra, N. (2008). PocketMatch: A new algorithm to compare binding sites in protein structures. *BMC Bioinformatics*, *9*, 1–17. https://doi.org/10.1186/1471-2105-9-543

Yu J, Zhou Y, Tanaka I, Yao M (2010) Roll: a new algorithm for the detection of protein pockets

and cavities with a rolling probe sphere. Bioinformatics 26(1):46–52

Zaliani, A., & Gancia, E. (1999). QSPR Studies. *J. Chem. Inf. Comput. Sci.*, *39*, 525–533.

Zhang, D., Zhao, Q., & Wu, B. (2015). Structural studies of G protein-coupled receptors. *Molecules and Cells*, *38*(10), 836–842. https://doi.org/10.14348/molcells.2015.0263

Zhang Y (2007) Template-based modeling and free modeling by I-TASSER in CASP7. Proteins. 69 Suppl 8:108-17. doi: 10.1002/prot.21702. PMID: 17894355.

Zhao, J., Deng, Y., Jiang, Z., & Qing, H. (2016). G protein-coupled receptors (GPCRs) in Alzheimer's disease: A focus on BACE1 related GPCRs. *Frontiers in Aging Neuroscience*, *8*(MAR), 1–15. https://doi.org/10.3389/fnagi.2016.00058

Zhou, H. X., & Pang, X. (2018). Electrostatic Interactions in Protein Structure, Folding, Binding, and Condensation. *Chemical Reviews*, *118*(4), 1691–1741. https://doi.org/10.1021/acs.chemrev.7b00305

Zhu H, Pisabarro MT (2011) MSPocket: an orientation-independent algorithm for the detection of ligand binding pockets. Bioinformatics 27(3):351–358

Appendices

APPENDIX A: TABLES

Table 11: Human conserved motifs of GPCR sequences across two IUPHAR families found by the MEME system

Ligands InChI Key	Sequence Region	Length of Region	GPCR UniProt ID	IUPHAR Family	Motif Starting Position in Full Sequence	Motif	E-value
YKMS	I_4	12	P14416	A	432	**EFRKAFLKILRC**	2.10E-04
		14	P35462	A	389		
		16	P21917	A	406		
		59	P43220	B	427		
FQUA	I_2	19	P41145	A	155	**DRYIAVAHPLKY**	5.90E-78
		20	P35372	A	166		
		21	P37288	A	148		
		20	O43194	A	132		
		21	P18825	A	148		
		20	P28222	A	146		
		21	P07550	A	130		
		21	P14416	A	131		
		20	P35367	A	124		
		20	P08172	A	120		
		20	P29274	A	101		
		21	O15552	A	106		
		20	P28335	A	151		
		21	O14843	A	112		
		20	P11229	A	122		
		20	P28223	A	172		
		20	P35368	A	142		
		20	P32245	A	146		
		19	Q99788	T2R	136		
		20	O14842	A	103		
		21	P24530	A	198		
		23	Q5NUL3	A	135		
		17	Q8TDU9	A	154		
		19	P59551	T2R	111		
	I_4	12	P14416	A	433	**FRRAFKKJLRC**	1.20E-30

Ligands InChI Key	Sequence Region	Length of Region	GPCR UniProt ID	IUPHAR Family	Motif Starting Position in Full Sequence	Motif	E-value
		24	P08172	A	447		
		17	P35367	A	475		
		28	P32245	A	309		
		19	P28222	A	376		
		169	P35368	A	355		
		109	O43194	A	351		
		53	P24530	A	393		
		84	P07550	A	332		
		122	P29274	A	295		
		87	P28335	A	375		
		21	P18825	A	444		
		87	P28223	A	387		
		39	P11229	A	425		
		28	Q9NYW1	T2R	289		
		62	P35372	A	345		
		99	P25103	A	312		
		16	Q9NYW5	T2R	287		
		47	P41145	A	337		
		45	Q5NUL3	A	326		
		67	O14843	A	283		
		54	O15552	A	280		
		22	P59536	T2R	290		
		211	Q8IZ08	A	327		
		35	Q9NYV8	T2R	287		
		13	P41145	A	123		
		11	P35372	A	134		
		20	O14843	A	80		
		13	P28222	A	114		
		17	P08172	A	88		
	E_2	17	Q8TDU9	A	106	**RWPLGRVLC**	4.80E-33
		10	P18825	A	116		
		16	P28223	A	140		
		12	P24530	A	166		
		22	O15552	A	74		
		17	Q99788	A	104		

Ligands InChI Key	Sequence Region	Length of Region	GPCR UniProt ID	IUPHAR Family	Motif Starting Position in Full Sequence	Motif	E-value
		17	P28335	A	119		
		15	P14416	A	99		
		15	O14842	A	71		
		18	P35367	A	92		
		11	P07550	A	98		
		17	P11229	A	90		
		14	Q5NUL3	A	103		
		20	P25103	A	97		
		22	Q9NYV7	T2R	71		
		15	P37288	A	116		
	H_3	25	P48546	B	224	**QYPIGANYAPLLVEG**	1.50E-06
		24	P47871	B	232		
		21	Q8IZ08	A	86		
		22	O14842	A	80		
XLWJ	E_1	162	Q03431	B	142	**GHVYRKCDANGSW**	5.50E-02
		116	P43220	B	98		
		577	Q9UBS5	C	75		
	I_2	20	Q6W5P4	A	145	**DRYHAITYPM**	7.60E-02
		19	P21728	A	120		
		20	P16473	A	518		
		25	O75899	C	576		
	H_7	21	Q6W5P4	A	322	**NSALNPIIYC**	5.10E-02
		25	P21728	A	323		
		21	P43220	B	394		
		23	Q03431	B	451		
		22	P16473	A	670		
NKOP	Full	463	P43220	B	211	**QHQWD**	4.20E-02
		1194	Q13255	C	907		
USZP	Full	463	P43220	B	356	**QEWWZHRRQC**	6.60E-01
		1212	P41594	C	37		
FQUA	E_4*	22	P35367	A	452	**CKEACNETLREAKLF**	5.50E-05
		25	P28335	A	337		
		26	P28223	A	349		
		23	P28222	A	340		

Ligands InChI Key	Sequence Region	Length of Region	GPCR UniProt ID	IUPHAR Family	Motif Starting Position in Full Sequence	Motif	E-value
		22	P47871	B	369		
		25	P48546	B	361		
		23	P18825	A	408		
		24	P11229	A	391		
BYBL	E_3*	35	P13945	A	185	**EARRCYNDPKCCDFASNMPY**	2.40E-03
		36	P08588	A	205		
		25	P43220	B	286		
DTZD	E_1*	561	Q14831	C	586	**WHLPWA**	8.10E-01
		121	P43220	B	87		

*We start the region 5 amino acids before and ended 5 amino acids after E_3, and E_4. Ended 5 amino acids after E_1

APPENDIX B: PROCEDURE FOR WORKFLOW

Files and codes in this procedure are publicly available at:
Note: The Folders contain data on different set of ligands, this is the case because some of the processes were automated during analysis of the control dataset. Folders containing codes and some data are publicly available at github.com/owusukd/GPCR_Ligand_Interaction
Note: Do this work sequentially from first page downwards. Also, make sure to change the working directory in all the codes appropriately.

Data Collection

Folder "Pipeline > GPCR_Ligand_Data"

1. Data were downloaded from GLASS, BindingDB, and IUPHAR.
    - *Sub-folder BindingDB*
        - On the website of BIndingDB, we go to download, and on the download page, under "All data in BindingDB", you download the zip tsv file with name "BindingDB_All_year-month-number.tsv.zip".
        - After downloading, we then clean the tsv file using R code "BindingDB_Data_Cleaning.R".
    - *Sub-folder GLASS*
        - On GLASS website (https://zhanggroup.org/GLASS/), we click on download and then download the "All interaction data in TSV format" data.
        - After downloading, we then clean the tsv file using R code "GLASS_Data_Cleaning.R".
    - *Sub-folder IUPHAR*

- On IUPHAR website (https://www.guidetopharmacology.org/download.jsp), we downloaded the "all interaction data for ligands and targets" and "complete ligand list" tsv files.
- We then merged the two data sets and cleaned it using the R code "IUPHAR_Data_Cleaning.R".

- GPCR-PEnDB
  - On the website (https://gpcr.utep.edu/advanced), we download data on GPCRs confirmed or predicted
  - Data is renamed as "GPCR_PEnDB.tsv"
  - The UniProt IDs are used to subset the combined data from GLASS, BindingDB, and IUPHAR to retain only GPCR-ligand interaction data. This is done when combining the data sets: next bullet point below
- Combining Data sets
  - Data from GLASS, IUPHAR, BindingDB and GPCR-PEnDB were combined and restructured using the R code "Combine_GLASS_IUPHAR_BindingDB_Data.R" and create the file "Final_Data.tsv"

2. GPCR sequence data were downloaded from GPCR-PEnDB
Folder "Pipeline> GPCR_Sequence_Data"

- On the website (https://gpcr.utep.edu/advanced), we download sequence data on GPCRs (confirmed or predicted)
  - For the GPRCs that bind the same ligand, we download their sequence data as one fasta file, e.g., "AJLFQFYMLRXVHV-UHFFFAOYSA-N.fasta"

3. Data on GPCR regional (N-terminal, extracellular loops, intracellular loops, the seven helices, and the C-terminal) positions
Folder "Pipeline> GPCR_Sequence_Data"

- We gathered data on the positions of the regions of the GPCRs from UniProt
  - That is the beginning and the ending of the regions
  - Eg. "AJLFQFYMLRXVHV-UHFFFAOYSA-N.txt"
- Data on the regional positions (e.g., AJLFQFYMLRXVHV-UHFFFAOYSA-N.txt) were used to cut the GPCR sequences into the respective regions for motif search
  - We save the sequences of GPCRs in one fasta file e.g., "AJLFQFYMLRXVHV-UHFFFAOYSA-N.fasta"
  - We used the R code "cutSequencesIntoPieces*.R"
    - *: there are different versions of the code for different type of cut
      - cutSequencesIntoPieces.R: for cutting the sequence into the different regions with no modifications including the N-terminus label as E1 and C-terminus as C4
      - cutSequencesIntoPieces_Ei.R: for cutting the sequences into the extracellular loops including the N-terminus label as E1

        - cutSequencesIntoPieces_Ei_add_5.R: for cutting the sequences into the modified extracellular loops by adding 5 more amino acid either at the end, beginning or both ends including the N-terminus label as E1

- We create the *sub-folder* e.g., "*fasta_MEME"* and *sub-sub-folder* e.g., "*AJLFQFYMLRXVHV-UHFFFAOYSA-N"* and copy the cut sequences into them e.g., "AJLFQFYMLRXVHV-UHFFFAOYSA-N_C1.txt"
    - Naming convention for the *sub-folder* e.g., "*fasta_MEME"*
        - *fasta_MEME*: means motif search was done using default settings
        - *fasta_MEME_full*: means motif search was done on the full sequence of the GPCRs
        - *fasta_MEME_3*: means motif search was done setting the min length of the motif to be 3
        - *fasta_MEME_3_10_mot*: motif search was done setting the min length of the motif to be 3 and retaining only 10 motifs
        - *fasta_MEME_add_5*: motif search was done on the modified regions of the extracellular loops including the N-terminus label as E1
        - *fasta_MEME_add_5_10_mot*: motif search was done on the modified regions of the extracellular loops including the N-terminus label as E1, and retaining only 10 motifs
        - *fasta_MEME_E_i*: means motif search was done on all the extracellular loops of all the GPCRs that bind the same ligand as one fasta file including the N-terminus label as E1
        - *fasta_MEME_E_i_10_mot*: means motif search was done on all the extracellular loops of all the GPCRs that bind the same ligand as one fasta file including the N-terminus label as E1, and retaining only 10 motifs
        - **The running of the motif search is done below on these sequence files**

4. GPCR 3D data
Folder "Pipeline >GPCR_3D_Data"

- We obtain a list of UniProt IDs of GPCRs confirmed or predicted from GPCR-PEnDB
    - gpcrpendb_results_1612200136.41.tsv: we saved the UniProt IDs as a separate file "GPCR_Pen.txt"
- With this list (GPCR_Pen.txt) we searched PDB for each one of them if there exist a 3D structure for it, using the advance search available on PDB
- From the search result, we selected PDB ID, experimental method, ligand ID, and Accession code(s) through the custom table option and downloaded the resulting csv file
    - Finding Unique GPCR_2021.csv

- Where multiple UniProt IDs are given in the data, we crosscheck if all the UniProt IDs are GPCRs and they have structures on PDB.
    - We used the code "rscbpdbscript.R"
- We then download the 3D structures of the GPCRs
    - Where there are multiple 3D structures on PDB we download the one with the longest sequence in the structure on PDB.
        - To do this, we display the search results in sequence form on PDB and select the one with longer sequence in the 3D structure and then wee download that 3D structure.

5. Ligand 3D structure
We downloaded the ligand SMILES and 3D structure on this website: https://pubchem.ncbi.nlm.nih.gov by searching for the ligand using the ligand **InChKey** (this is part of the Combined data "Final_Data.tsv")

Folder "Pipeline > Ligands"

- We downloaded the 3D structures of the ligands by searching for the ligands on google using the InChI Key of the ligands
    - The structures are converted to have the file extension .pdbqt using the appropriate script from the list below:
        - mol2_to_pdbqt.sh
        - sdf_to_pdbqt.sh
- For ligands with no available 3D structure to download, we converted the SMILES of the ligand into 3D structures using Open Babel
    - We copy and save the SMILES in a text file with the file extension .smi
        - E.g., AJLF.smi
    - We then used the script below to convert the smiles into a 3D structure
        - convert_ligand_smiles_into_3D.sh"

Analysis on GPCR Ligand Data
Folder "Pipeline > GPCR_Ligand_Data"

6. We performed analysis on the combine data (IN PAGE 1) to determine ligands which bind GPCRs of different families.
- We used the R code "Analysis_lig_mult_GPCR.R" in folder "*Pipeline*"

Sequence Mofit Search
Folder "Entire_work_organized > GPCR_Sequence_Data"

7. For GPCRs that bind the same ligand:
Sequence data needed for this section is under **Data Collection above page 1 and 2: the full and the cut sequences**
- We save the sequences of the GPCRs in one fasta file
- We performed a motif search on the full sequences of the GPCRs
- We performed a motif search on the regional sequences of the GPCRs
- We performed a motif search on a modified regional sequence of the GPCRs

    - Modified: meaning we cut the regions adding 5 amino acids before the actual start of the regional sequence and/or end the regional sequence five amino acids after the actual end of the regional sequence where they are possible
- Motifs that had E-value < 0.1 were retained as significant motifs (for the manuscript we only retained motifs that had E-value < 0.01)
    - We create a table of the ligands and the GPCRs they bind to and add the significant motifs as a column on the table manually (see Result Section of Dissertation under Motif Search and Appendix)
- We used the scripts "run_meme*.sh"
    - *: there are different versions of the script running different MEME
    - Naming convention for the scripts
        - *run_meme.sh*: means motif search was done using default settings
        - *run_meme_3.sh*: means motif search was done setting the min length of the motif to be 3
        - *run_meme_3_10_mot.sh*: motif search was done setting the min length of the motif to be 3 and retaining only 10 motifs
        - *run_meme_full.sh*: means motif search was done on the full sequence of the GPCRs
        - *run_meme_add_5.sh*: motif search was done on the modified regions of the extracellular loops including the N-terminus label as E1
        - *run_meme_add_5_10.sh*: motif search was done on the modified regions of the extracellular loops including the N-terminus label as E1, and retaining only 10 motifs
        - *run_meme_E_i.sh*: means motif search was done on all the extracellular loops of all the GPCRs that bind the same ligand as one fasta file including the N-terminus label as E1
        - *run_meme_E_i_10_mot.sh*: means motif search was done on all the extracellular loops of all the GPCRs that bind the same ligand as one fasta file including the N-terminus label as E1, and retaining only 10 motifs

Binding Pocket Prediction and Comparison

Pockets are predicted before the pocket comparisons are done.
Folder "Pipeline > Binding_pocket_prediction_and_comparison"

Binding Pocket Prediction

8. For the GPCRs that bind the same ligand:
Folder "Pocket_Predictions"

- If the binding site of the ligand is unknown:
    - First, we clean the GPCR PDB files
        - We used "clean_PDB_files.sh"
    - We predict binding pockets for the GPCR (for GPCR files with file extension .pdb e.g., "0HK_P08173_A_5dsg.pdb")
        - We used p2rank_2.2

        - We used the code "run_P2rank.sh"
    - We extract the pockets from the GPCR PDB files e.g., 0HK_P08173_A_5dsg_pkt_1.txt
        - These files are copied to the sub-folder created for each of the ligands e.g., "*0HK*" under the folder "*Pipeline > Binding_pocket_prediction_and_comparison > APoc*"
        - We used the code "get_pkt_AA_coordinates.py" to extract the pockets. The code requires the predicted pocket numbers.
            - This code also creates docking configuration files needed later for GPCR ligand docking
            - The docking configuration files e.g., "0HK_P08173_A_5dsg_config.txt" are copied into the folders created for each of the ligands under the folder *"Pipeline > AutoDock_ligands_Proteins > Docked"* e.g., "*0HK*"

- If the binding site of the ligand is known, we only save the pocket (as in the case of the control data)
    - To do this:
        - We predict binding pockets for the GPCR with the ligand in bound with it
        - We then go to each of the ligand folder which contains the results for the pocket prediction and then go into the sub-folder "*visualizations*" and open the files with file extension .pml (e.g., 0HK_P08173_A_5dsg.pdb.pml) with PyMol
        - In the open PyMol window, we check which pocket contains the ligand and we note the pocket number
            - The pocket number is used in the code "get_pkt_AA_coordinates.py" to extract the pocket and create the docking configuration files needed later for GPCR ligand docking
            - The docking configuration files e.g., "0HK_P08173_A_5dsg_config.txt" are copied into the folders created for each of the ligands under the folder *"Pipeline > AutoDock_ligands_Proteins > Docked"* e.g., "*0HK*"

Binding Pocket Comparison

9. For the GPCRs that bind the same ligand:

Folder "APoc"

- We create folders for each of the ligands and copy the 3D structures of the GPCRs they bind to into it and also the pocket files for them e.g., "0HK_P08173_A_5dsg_pkt_1.txt" (from *Binding Pocket Prediction* above)
- For the binding site of the ligand (whether known or predicted):
    - We add the pocket files of a GPCR to the GPCR's 3D structure file
        - We used the code "add_pkt_to_protein_file.py" to do that

- We perform a pairwise comparisons of the predicted binding pockets across the GPCRs
  - We used the code "run_APoc.sh" to do that
  - "run_APoc.sh" will produce result files e.g., "0HK_P08173_A_5dsg.pdb_vs_0HK_P11229_A_5cxv.pdb_pocket_compare_results.txt"
- We then combine the results from the comparison
  - We used the code "parse_bind_poc_comp_results.py" to do that
  - "parse_bind_poc_comp_results.py" requires the result files e.g., "0HK_P08173_A_5dsg.pdb_vs_0HK_P11229_A_5cxv.pdb_pocket_compare_results.txt"
  - "Combine_pocket_comp_results.tsv" will be generated as output

3D Structural Comparison

Folder "Pipeline > 3D_Comparison_Pocket_overlap_scores "

10. For GPCRs that bind the same ligand:

Folder "3D_structural_comparison"

- We create sub-folders for each of the ligands e.g., "*0HK*" and copy the cleaned 3D structures of the GPCRs with file extension .pdb e.g., "0HK_P08173_A_5dsg.pdb" (from *Binding Pocket Prediction* above) they bind to into it
- We performed pairwise 3D structural comparison of the GPCRs using FATCAT
  - This comparison is done both considering flexibility (allowing twist) and rigidity (not allowing twist)
  - We used the code "run_FATCAT.sh" and it does both flexible and rigid case
  - "run_FATCAT.sh" requires the 3D structures of the GPCRs e.g., "0HK_P08173_A_5dsg.pdb"
  - "run_FATCAT.sh" produces the alignment files with file extension .aln e.g., "0HK_P08173_A_5dsg_0HK_P11229_A_5cxv_flex.aln"
- We saved the RMSD score from the comparison and also, we also save the superimposed parts of the GPCRs (that is, the parts of the two GPCRs under comparison which were found to be 3D structurally similar) in .pdb format
  - We used the code "get_superimposed_3D_parts.py" to get the RMSD and the superimposed parts
  - "get_superimposed_3D_parts.py" requires the alignment files
  - "get_superimposed_3D_parts.py" produces the files e.g., "0HK_P08173_A_5dsg_flex_with_0HK_P11229_A_5cxv.txt" in the ligand folder e.g., "*0HK*" and "3D_Similar_RMSD.tsv" in the folder *"Pipeline > 3D_Comparison_Pocket_overlap_scores > 3D_structural_comparison"*

11. Overlap score:

Folder "Overlap_scores"

- We create sub-folders for each of the ligands e.g., "*0HK*" and copy the superimposed files e.g., "0HK_P08173_A_5dsg_flex_with_0HK_P11229_A_5cxv.txt" (from *3D structure comparison* above) and the pocket files e.g., "0HK_P08173_A_5dsg_pkt_1.txt" (from *Binding Pocket Prediction* above)

- After the superimposed parts of the GPCRs compared have been saved e.g., "0HK_P08173_A_5dsg_flex_with_0HK_P11229_A_5cxv.txt"
    - We used each to compare with the binding pocket(s) of their GPCRs e.g., "0HK_P08173_A_5dsg_pkt_1.txt"
    - A score is calculated for this comparison both for flexible case and the rigid case
        - We used the code "scoring_code_control_data.py"
    - An average score is calculated for the flex and rigid cases for each GPCR We then sum the averages for a pair of GPCRs compared
        - This is done at under **Analysis of Results below**
        - We used the code "Combine_Result_Data_Analysis.R" in the folder "*Pipeline*"

GPCR Ligand Docking

Folder "Pipeline > AutoDock_ligands_Proteins"

12. For GPCRs that bind the same ligand:

- We create sub-folders for each of the ligands e.g., "*0HK*" and copy the docking configuration files e.g., "0HK_P08173_A_5dsg_config.txt" and the 3D structures of the GPCRs with file extension .pdbqt e.g., "0HK_P08173_A_5dsg.pdbqt" (all from *Binding Pocket Prediction* above) into it
- We dock the ligands into the binding pocket(s)
    - We first prepare the GPCR pdb file using AutoDock MGL tools
        - Kollman charges were added
        - Charge Field was set to Kollman
        - AD4 type was assigned, and the file was saved as a pdbqt file
        - First three points are done during cleaning of the PDB files under *Binding Pocket Prediction* above
    - We used the .pdbqt files of the ligands for docking
        - We copy the ligand folder *"Pipeline > Ligands > Control_Data_Ligands"* into the folder*"Pipeline > AutoDock_ligands_Proteins"*
    - We then dock the ligand into the pocket(s) of each GPCR using "run_vina.sh"
        - "run_vina.sh" requires the .pdbqt files of the GPCRs and the ligands
        - "run_vina.sh" produces the log files e.g., "0HK_P08173_A_5dsg_log.txt" which contains the docking results, and the docked ligand files e.g.,

"0HK_P08173_A_5dsg_out.pdbqt" which contains the ligand in a docked pose and conformation
        - The docked ligand files e.g., "0HK_P08173_A_5dsg_out.pdbqt" are copied into the "*Pipeline > Ligand_Pose_Comformation_Docked*" for **Ligand Binding Pose and Conformation analysis below**
    - The result from the docking e.g., "0HK_P08173_A_5dsg_log.txt" are gathered together
        - We used the code "parse_dock_results.py" to do that
        - "parse_dock_results.py" requires the docking log files which contains the docking results e.g., "0HK_P08173_A_5dsg_log.txt"
        - "parse_dock_results.py" produces the file "AutoDock_vina_Results.tsv" under the folder *"Pipeline > AutoDock_ligands_Proteins"*

Ligand Binding Pose and Conformation

Folder "Pipeline > Ligand_Pose_Comformation_Docked"

13. For GPCRs that bind the same ligand:

- We compare the conformation of the ligands after docking for each pocket compared
    - We used the code "ligs_align_docked.py"
        - "ligs_align_docked.py" requires the docked ligand files e.g., "0HK_P08173_A_5dsg_out.pdbqt" which contains the ligand in a docked pose and conformation
        - "ligs_align_docked.py" produces the "Ligs_Align_Pkt_Docked.tsv"

Pockets Electrostatic Properties

14. For GPCRs that bind the same ligand:

Folder "Pipeline > 3D_Comparison_Pocket_overlap_scores "

- We calculate electrostatic properties of each pair of pockets compared from each of the GPCRs compared
    - We first get the three letter code of the amino acids in the pockets using "get_AA_pkt.py" in the folder "3D_Comparison_Pocket_overlap_scores"
        - "get_AA_pkt.py" requires pocket files e.g., "0HK_P08173_A_5dsg_pkt_1.txt" in the sub-folders of *"Pipeline > 3D_Comparison_Pocket_overlap_scores > Overlap_scores"*
        - "get_AA_pkt.py" produces the file "AA_Pkt.xlsx" in the folder *"Pipeline > 3D_Comparison_Pocket_overlap_scores"*
    - We then used the code "Combine_Result_Data_Analysis.R" in the folder "*Pipeline*" to calculate the Molecular Surface Weighted Holistic Invariant Molecular (MS-WHIM) scores for each pockets compared
        - This is done at under **Analysis of Results below**

- We then calculate Chebyshev distance between the MS-WHIM scores for each pair of pockets compared
    - We then used the code "Combine_Result_Data_Analysis.R" in the folder "*Pipeline*"
        - This is done at under **Analysis of Results below**

Protein Ligand Interaction

Folder "Pipeline > Protein_Ligand_Interaction_Actual > Actual"

15. For GPCRs that bind the same ligand:

- We save the docked ligand and the GPCR 3D structure as one .pdb file
    - This is done by opening the docked ligand files e.g., "0HK_P08173_A_5dsg_out.pdbqt", and the GPCR it was docked into e.g., "0HK_P08173_A_5dsg.pdbqt" in one PyMol window
        - Then we export molecule as a PDB file e.g., "0HK_P08173_A_5dsg.pdb"
    - This is done for all the pocket(s) we docked the ligand into

- We then run LigPlot+ to determine the amino acids of the GPCR that interacts with the ligand
    - First install LigPlot+ from https://www.ebi.ac.uk/thornton-srv/software/LigPlus/download.html
    - Then we generate the interactions for all the pocket(s) we docked the ligand into
    - We save the three letter code of the amino acids involved in a hydrogen bond with the ligand for each pair of pockets compared in separate columns (see the file "Ligs_Protein_Interaction.xlsx")
    - We used the code "Combine_Result_Data_Analysis.R" in the folder "*Pipeline*" to perform analysis on the number of same residues across the pockets we are comparing that interact with the ligand
        - This is done at under **Analysis of Results below**

Analysis of Results

16. All the data generated from the running of the pockets comparison, ligand conformation analysis, docking results, and protein ligand interaction were combined for further analysis: some manually and some using the code "Combine_Result_Data_Analysis.R" in the folder "*Pipeline*"

- We used the combined data of results to generate other features
- We then performed Pearson correlation analysis between PS-score and other features
- We used the code "Combine_Result_Data_Analysis.R" in the folder "*Pipeline*"

APPENDIX C: DATA COLLECTION AND ANALYSIS

C.1: BindingDB Data

```
bindingDb.all <- read.csv(file = "BindingDB_All.tsv", sep = '\t',
                          fill = T, stringsAsFactors = F, header = T)

colnames(bindingDb.all)
# Columns to retain
names(bindingDb.all)[c(42)] <- c("UniProt.ID.of.Target")
bindingDb.all <- bindingDb.all[, c(2,4,9:12,27,28,33,39,42)]
cond <- which(bindingDb.all$UniProt.ID.of.Target == "")
if(length(cond) != 0){
  bindingDb.all <- bindingDb.all[-cond,]
}

names(bindingDb.all) <- c("Ligand.SMILES","Ligand.InChI.Key",
"Ki.nM","IC50.nM","Kd.nM","EC50.nM","Ligand.HET.ID.in.PDB",
"PDB.ID.for.Ligand.Target.Complex","DrugBank.ID.of.Ligand","PDB.ID.of.Target.
Chain","UniProt.ID.of.Target")

write.table(bindingDb.all, sep = "\t",
          file =
"/Users/kwabena/Research/GPCR/Entire_work_organized/GPCR_Ligand_Data/BindingD
B/bindingDb.all.tsv",
          row.names = FALSE)

remove(list = ls(all.names = T))
```

C.2: GLASS Data

```
glass.all <- read.csv(file = "interactions_total.tsv", sep = '\t',
                          fill = T, stringsAsFactors = F, header = T)

colnames(glass.all)
# Columns to retain
glass.all <- glass.all[, c(1:6)]
cond <- which(glass.all$UniProt.ID == "")
if(length(cond) != 0){
  glass.all <- glass.all[-cond,]
}

names(glass.all) <- c("UniProt.ID.of.Target","Ligand.InChI.Key","Parameter",
                      "Value","Unit","Database.Source")

write.table(glass.all, sep = "\t",
          file =
"/Users/kwabena/Research/GPCR/Entire_work_organized/GPCR_Ligand_Data/GLASS/gl
ass.all.tsv",
          row.names = FALSE)

remove(list = ls(all.names = T))
```

C.3: IUPHAR Data

```
interactions <- read.csv(file = "interactions.tsv", sep = '\t', fill = F,
stringsAsFactors = F,
                         header = T, skip = 1)
colnames(interactions)
if(colnames(interactions)[16] == "X.Ligand.ID"){
  names(interactions)[16] <- "Ligand.ID"
}

ligands <- read.csv(file = "ligands.tsv", sep = '\t',
                    fill = F, stringsAsFactors = F, header = T, skip = 1)
colnames(ligands)

iuphar.all <- merge(interactions, ligands, by.x = "Ligand.ID", by.y =
"Ligand.id")

colnames(iuphar.all)

iuphar.all <- iuphar.all[, c(6,29,30,31,32,55,56)]

write.table(iuphar.all, sep = "\t",
          file =
"/Users/kwabena/Research/GPCR/Entire_work_organized/GPCR_Ligand_Data/IUPHAR/i
uphar.all.tsv",
          row.names = F)

remove(list = ls(all.names = T))
```

C.4: Combining BindingDB, GLASS, and IUPHAR Data Sets

```
# get data
GPCR_Pen <-read.csv("GPCR_PEnDB.tsv",header = T, fill=F, sep = "\t", skip =
3,
                    stringsAsFactors = F)
GPCR_Pen1 <- GPCR_Pen[,c(1,11)]
names(GPCR_Pen1) <- c("UniProt.ID.of.Target","Target.Common.name")

GPCR_Pen <-as.data.frame(GPCR_Pen)
colnames(GPCR_Pen)
GPCR_Pen <- GPCR_Pen[,c(1,12)]
names(GPCR_Pen) <- c("UniProt.ID.of.Target","IUPHAR.Class.of.Target")

bindingDb.all <- read.csv(file = "bindingDb.all.tsv", sep = '\t',
                          fill = T, stringsAsFactors = F, header = T)

iuphar.all <- read.csv(file = "iuphar.all.tsv", sep = '\t', fill = F,
stringsAsFactors = F,
                       header = T)

glassData <- read.csv(file = "glass.all.tsv", sep = '\t', fill = F,
stringsAsFactors = F,
                      header = T)

colnames(bindingDb.all)
colnames(iuphar.all)
```

```
colnames(glass.all)

# check for the GPCRs
cond <- bindingDb.all$UniProt.ID.of.Target %in% GPCR_Pen$UniProt.ID.of.Target
sum(cond)
bindingDb.all <- bindingDb.all[cond, ]

cond <- glass.all$UniProt.ID.of.Target %in% GPCR_Pen$UniProt.ID.of.Target
sum(cond)
glass.all <- glass.all[cond, ]

cond <- iuphar.all$Target.UniProt.ID %in% GPCR_Pen$UniProt.ID.of.Target
sum(cond)
iuphar.all <- iuphar.all[cond, ]
names(iuphar.all)[c(1,6,7)] <- c("UniProt.ID.of.Target", "Ligand.SMILES",
"Ligand.InChI.Key")

colnames(bindingDb.all)
colnames(iuphar.all)
colnames(glass.all)

# get them to have the same columns
bindingDbNewNames <-
c("Affinity.Units","Affinity.High","Affinity.Median","Affinity.Low",
                        "Parameter","Value","Unit")
bindingDb.all[, bindingDbNewNames] <- NA
bindingDb.all[["Database.Source"]] <- "BindingDB"

glassNewNames <-
c("Ligand.SMILES","Ki.nM","IC50.nM","Kd.nM","EC50.nM","Ligand.HET.ID.in.PDB",

"PDB.ID.for.Ligand.Target.Complex","DrugBank.ID.of.Ligand","PDB.ID.of.Target.
Chain",

"Affinity.Units","Affinity.High","Affinity.Median","Affinity.Low")
glass.all[, glassNewNames] <- NA
glass.all[["Database.Source"]] <- "GLASS"

iupharNewNames <-
c("Ki.nM","IC50.nM","Kd.nM","EC50.nM","Ligand.HET.ID.in.PDB",

"PDB.ID.for.Ligand.Target.Complex","DrugBank.ID.of.Ligand","PDB.ID.of.Target.
Chain",
                    "Parameter","Value","Unit")
iuphar.all[, iupharNewNames] <- NA
iuphar.all[["Database.Source"]] <- "IUPHAR"


# check if they have same columns
colnames(bindingDb.all)
colnames(iuphar.all)
colnames(glass.all)

# get the columns in the same order
ord_glass <- c(1,2,7,3,4,5,8,9,10,11,17,18,19,16,12,14,13,15,6)
ord_iuphar <- c(1,7,6,16,17,18,8,9,10,11,3,4,5,2,12,14,13,15,19)
ord_bindingDb <- c(11,2,1,16,17,18,3,4,5,6,13,14,15,12,7,9,8,10,19)
bindingDb.all <- bindingDb.all[, ord_bindingDb ]
```

```
iuphar.all <- iuphar.all[, ord_iuphar]
glass.all <- glass.all[, ord_glass]

# append them together
Final_Data <- rbind(iuphar.all, bindingDb.all)
Final_Data <- rbind(Final_Data, glass.all)

# check and keep only GPCR data
cond <- Final_Data$UniProt.ID.of.Target == ""
sum(cond)
Final_Data <- Final_Data[!cond, ]

cond <- Final_Data$UniProt.ID.of.Target %in% GPCR_Pen$UniProt.ID.of.Target
sum(cond)
Final_Data <- Final_Data[cond, ]

# Checking for duplicates
suppressPackageStartupMessages(library(dplyr))
Final_Data <- distinct(Final_Data)

# ligands with no InChIKey
cond <- Final_Data$Ligand.InChI.Key == ""
sum(cond)
Final_Data <- Final_Data[-cond, ]

Final_Data <- merge(Final_Data, GPCR_Pen, by.x = "UniProt.ID.of.Target",
                    by.y = "UniProt.ID.of.Target")
cond <- Final_Data$IUPHAR.Class.of.Target =""
sum(cond)


##############################################################################
##########
# Restructure Final_Data set to have all affinity parameters on a ligand-GPCR
on one row
##############################################################################
##########

# keep only Set 1: Top 7 highest (Ki, IC50, Potency, EC50, Inhibition,
Activity, Kd, pKB,
# pKi, pIC50, pKd, pEC50) parameter
table(Final_Data$Parameter)
table(Final_Data$Affinity.Units)
table(Final_Data$Unit)

## combine some the parameters
# p[A50] = p[A]50 = pEC50
cond <- (Final_Data$Parameter == "p[A50]")|(Final_Data$Parameter ==
"p[A]50")|
  (Final_Data$Parameter == "pEC50")
sum(cond, na.rm = T)
Final_Data$Parameter[cond] <- "pEC50"
# -Log KD = pKD
cond <- (Final_Data$Parameter == "pKD")|(Final_Data$Parameter == "-Log KD")
sum(cond, na.rm = T)
Final_Data$Parameter[cond] <- "pKd"
# pKi(uM) = pKi
cond <- (Final_Data$Parameter == "pKi(uM)")|(Final_Data$Parameter == "pKi")
```

```
sum(cond, na.rm = T)
Final_Data$Parameter[cond] <- "pKi"
# pKB = PKb = PkB = pkB = pKb = -Log K B = -Log KB
cond <-
(Final_Data$Parameter=="pKb")|(Final_Data$Parameter=="pKB")|(Final_Data$Param
eter=="PKb")|

(Final_Data$Parameter=="PkB")|(Final_Data$Parameter=="pkB")|(Final_Data$Param
eter == "-Log KB")|
  (Final_Data$Parameter == "-Log K B")
sum(cond, na.rm = T)
Final_Data$Parameter[cond] <- "pKB"
# Ki = KI_MICROM
cond <- (Final_Data$Parameter == "KI_MICROM")|(Final_Data$Parameter == "Ki")
sum(cond, na.rm = T)
Final_Data$Parameter[cond] <- "Ki"

# subset Final_Data for data from glass, iuphar, bindingdb
cond <- Final_Data$Database.Source == "GLASS"
sub.glass <- Final_Data[cond,]
cond <- Final_Data$Database.Source == "IUPHAR"
sub.iuphar <- Final_Data[cond,]
cond <- Final_Data$Database.Source == "BindingDB"
sub.bindingdb <- Final_Data[cond,]

# remove duplicates in glass data, iuphar, and bindingdb
sub.glass <- distinct(sub.glass)
sub.iuphar <- distinct(sub.iuphar)
sub.bindingdb <- distinct(sub.bindingdb)

cond <- (sub.glass$Parameter == "Ki")|(sub.glass$Parameter == "IC50")|
  (sub.glass$Parameter == "Potency")|(sub.glass$Parameter == "EC50")|
  (sub.glass$Parameter == "Inhibition")|(sub.glass$Parameter == "Activity")|
  (sub.glass$Parameter == "Kd")|(sub.glass$Parameter == "pKB")|
  (sub.glass$Parameter == "pKi")|(sub.glass$Parameter == "pIC50")|
  (sub.glass$Parameter == "pKd")|(sub.glass$Parameter == "pEC50")
sum(cond, na.rm = T)
sub.glass <- sub.glass[cond,]

# keep units nM(Ki, Kd, IC50, EC50, Potency), %(Activity, Inhibition), -(pKB)
table(sub.glass$Unit, sub.glass$Parameter)

cond <- (sub.glass$Unit == "nM")|(sub.glass$Unit == "%")|(sub.glass$Unit ==
"-")
sum(cond, na.rm = T)
sub.glass <- sub.glass[cond,]

# keep only pKi, pIC50, pKd, pEC50, pKB of Affinity.Units
cond <-
(sub.iuphar$Affinity.Units=="pKi")|(sub.iuphar$Affinity.Units=="pIC50")|
  (sub.iuphar$Affinity.Units=="pKd")|(sub.iuphar$Affinity.Units=="pEC50")|
  (sub.iuphar$Affinity.Units=="pKB")
sum(cond, na.rm = T)
sub.iuphar <- sub.iuphar[cond,]

#### Glass
colnames(Final_Data)
table(sub.glass$Parameter)
```

```
table(sub.iuphar$Affinity.Units)
table(sub.bindingdb)
# create the variables Activity, Inhibition, Potency, pKB, pKi, pIC50, pEC50,
pKd
var.names <-
c("Activity.%","Inhibition.%","Potency.nM","pKB","pKi","pIC50","pEC50","pKd")
sub.glass[,var.names] <- NA
colnames(sub.glass)

var_indx <- c(21,22,23,7,8,10,9,24,27,25,26,28)
parameters <- c("Activity", "Inhibition", "Potency", "Ki", "IC50", "EC50",
"Kd", "pKB",
                "pEC50", "pKi", "pIC50", "pKd")
for (i in 1:length(parameters)){
  cond <- which((sub.glass$Parameter == parameters[i]), arr.ind = T)
  sub.glass[cond,var_indx[i]] <- sub.glass$Value[cond]
}

#### IUPHAR
sub.iuphar[,var.names] <- NA
parameters <- c("pKB","pEC50","pIC50","pKd","pKi")
colnames(sub.glass)

var_indx <- c(24,27,26,28,25)
for (i in 1:length(parameters)){
  cond <- which((sub.iuphar$Affinity.Units == parameters[i]), arr.ind = T)
  sub.iuphar[cond,var_indx[i]] <- paste(sub.iuphar$Affinity.Low[cond],
                                        sub.iuphar$Affinity.High[cond], sep = " -
")
}

#### Bindingdb
sub.bindingdb[,var.names] <- NA

# Rearrange column
Final_Data <- rbind(sub.bindingdb,sub.iuphar,sub.glass)
Final_Data <- Final_Data[,-c(4,5,6,11,12,13,14)]
colnames(Final_Data)
var.nm <- c(1,11,13,2,3,9,8,4:7,14:20,10,12)
Final_Data <- Final_Data[, var.nm]
colnames(Final_Data)

######### Separate operators from affinity values
library(stringr)
operator <- c("<" , ">", "=")
Final_Data$Affinity_relation <- NA

### Ki.nM do for "<" , ">", "="
for (op in operator) {
  symb <- str_which(Final_Data$Ki.nM, op)
  if (length(symb) >= 1){
    ki <- Final_Data$Ki.nM[symb]
    ki <- unlist(ki)
    ki_List <- strsplit(ki,split= op, fixed=T)

    ki.val <- data.frame(stringsAsFactors = F)
    for (q in 1:length(ki_List)) {
      ki.val.list <- data.frame(val = ki_List[[q]][2], stringsAsFactors = F)
```

```
      ki.val <- rbind(ki.val, ki.val.list, stringsAsFactors = F)
    }
    ki.val <- as.vector(unlist(ki.val))

    Final_Data$Ki.nM[symb] <- ki.val
    Final_Data$Affinity_relation[symb] <- op

    rm(ki_List, ki.val, ki.val.list,symb,ki,q)
  }
}

### Kd.nM do for "<" , ">", "="
for (op in operator) {
  symb <- str_which(Final_Data$Kd.nM, op)
  if (length(symb) >= 1){
    kd <- Final_Data$Kd.nM[symb]
    kd <- unlist(kd)
    kd_List <- strsplit(kd,split=op, fixed=T)

    kd.val <- data.frame(stringsAsFactors = F)
    for (q in 1:length(kd_List)) {
      kd.val.list <- data.frame(val = kd_List[[q]][2], stringsAsFactors = F)
      kd.val <- rbind(kd.val, kd.val.list, stringsAsFactors = F)
    }
    kd.val <- as.vector(unlist(kd.val))

    Final_Data$Kd.nM[symb] <- kd.val
    Final_Data$Affinity_relation[symb] <- op

    rm(kd_List, kd.val, kd.val.list,symb,kd,q)
  }
}

### IC50.nM do for "<" , ">"
for (op in operator) {
  symb <- str_which(Final_Data$IC50.nM, op)
  if (length(symb) >= 1){
    IC50 <- Final_Data$IC50.nM[symb]
    IC50 <- unlist(IC50)
    IC50_List <- strsplit(IC50,split= op, fixed=T)

    IC50.val <- data.frame(stringsAsFactors = F)
    for (q in 1:length(IC50_List)) {
      IC50.val.list <- data.frame(val = IC50_List[[q]][2], stringsAsFactors =
F)
      IC50.val <- rbind(IC50.val, IC50.val.list, stringsAsFactors = F)
    }
    IC50.val <- as.vector(unlist(IC50.val))

    Final_Data$IC50.nM[symb] <- IC50.val
    Final_Data$Affinity_relation[symb] <- op

    rm(IC50_List, IC50.val, IC50.val.list,symb,IC50,q)
  }
}

### EC50.nM do for "<" , ">", "="
for (op in operator) {
```

```
  symb <- str_which(Final_Data$EC50.nM, op)
  if (length(symb) >= 1){
    EC50 <- Final_Data$EC50.nM[symb]
    EC50 <- unlist(EC50)
    EC50_List <- strsplit(EC50,split=op, fixed=T)

    EC50.val <- data.frame(stringsAsFactors = F)
    for (q in 1:length(EC50_List)) {
      EC50.val.list <- data.frame(val = EC50_List[[q]][2], stringsAsFactors =
F)
      EC50.val <- rbind(EC50.val, EC50.val.list, stringsAsFactors = F)
    }
    EC50.val <- as.vector(unlist(EC50.val))

    Final_Data$EC50.nM[symb] <- EC50.val
    Final_Data$Affinity_relation[symb] <- op

    rm(EC50_List, EC50.val, EC50.val.list,symb,EC50,q)
  }
}

# make some variables numeric
Final_Data$Ki.nM <- as.numeric(Final_Data$Ki.nM)
Final_Data$Kd.nM <- as.numeric(Final_Data$Kd.nM)
Final_Data$IC50.nM <- as.numeric(Final_Data$IC50.nM)
Final_Data$EC50.nM <- as.numeric(Final_Data$EC50.nM)

cond <- is.na(Final_Data$Affinity_relation)
sum(cond, na.rm = T)
Final_Data$Affinity_relation[cond] <- "="

# Rearrange columns
colnames(Final_Data)
Final_Data <- Final_Data[,c(1:7,21,8:20)]

Final_Data <- merge(Final_Data, GPCR_Pen1, by.x = "UniProt.ID.of.Target",
                    by.y = "UniProt.ID.of.Target", suffixes = c("",".y"))
Final_Data <- Final_Data[,c(1,22,2:21)]
cond <- Final_Data$Ligand.InChI.Key == ""
sum(cond, na.rm = T)
Final_Data <- Final_Data[!cond,]

# write data to tsv file
write.table(Final_Data, sep = "\t",
            file =
"/Users/kwabena/Research/GPCR/Entire_work_organized/GPCR_Ligand_Data/Final_Da
ta.tsv",
            row.names=FALSE)

rm(list = ls())
```

C.5: Analysis of GPCR Ligand Interaction Data

```
#####################################################################
# Get Ligands that bind to multiple GPCRs of different families
#####################################################################
library(foreach)
```

```
library(doParallel)
library(dplyr)


Final_Data <- read.table(file = "Final_Data.tsv", sep = "\t", fill = F,
stringsAsFactors = F,                        header = T)
# Find unique ligands
ligand_uniq <- as.vector(unique(Final_Data$Ligand.InChI.Key))
l.na <- which(is.na(ligand_uniq), arr.ind = T)
if (sum(l.na) > 0){
  ligand_uniq <- ligand_uniq[-c(l.na)]
}

# Parallel approach to finding ligands that binds to GPCRs of multiple
families
# We save the data
cores=detectCores()
cl <- makeForkCluster(cores[1]-4) #not to overload the computer
registerDoParallel(cl)

ligand_mult_GPCR_fam <- foreach(i=1:length(ligand_uniq), .combine = rbind)
%dopar% {
  cond <- Final_Data$Ligand.InChI.Key == ligand_uniq[i]
  if (sum(cond, na.rm = T) > 1){
    subs <- Final_Data[cond,c(1,2,4,5)]
    subs <- distinct(subs)
    len <- length(unique(subs$IUPHAR.Class.of.Target))
    if (len > 1){
      subs
    }
  }
}
stopCluster(cl)

# write data to tsv file
write.table(ligand_mult_GPCR_fam, sep = "\t",
            file
="/Users/kwabena/Research/GPCR/Entire_work_organized/ligand_mult_GPCR_fam_Dat
a.tsv",
            row.names=FALSE)

cond <- ligand_mult_GPCR_fam$Target.Common.name == "Human"
sum(cond, na.rm = T)
ligand_mult_GPCR_fam <- ligand_mult_GPCR_fam[cond,]
ligand_uniq <- as.vector(unique(ligand_mult_GPCR_fam$Ligand.InChI.Key))
rm(cond)

cores=detectCores()
cl <- makeForkCluster(cores[1]-4) #not to overload the computer
registerDoParallel(cl)

ligand_mult_GPCR_fam_Human <- foreach(i=1:length(ligand_uniq), .combine=
rbind) %dopar% {
  cond <- ligand_mult_GPCR_fam$Ligand.InChI.Key == ligand_uniq[i]
  if (sum(cond, na.rm = T) > 1){
    subs <- ligand_mult_GPCR_fam[cond,]
    subs <- distinct(subs)
    len <- length(unique(subs$IUPHAR.Class.of.Target))
```

```
    if (len > 1){
      subs
    }
  }
}

stopCluster(cl)

# write data to tsv file
write.table(ligand_mult_GPCR_fam_Human, sep = "\t",
                  file =
"/Users/kwabena/Research/GPCR/Entire_work_organized/ligand_mult_GPCR_fam_Huma
n_Data.tsv",
                  row.names=FALSE)

rm(list = ls())
```

C.6: Ligand 3D Structures

C.6.1: Convert from mol2 to pdbqt

```
#! /bin/bash
# convert files from .mol2 to .pdbqt
for f in ./*.mol2; do
      b=`basename $f .mol2`
      echo Processing ligand $b
      obabel -i mol2 $f -o pdbqt -O $b.pdbqt -xh --partialcharge gasteiger
done
```

C.6.2: Convert from sdf to pdbqt

```
#! /bin/bash
# convert files from .mol2 to .pdbqt
for f in ./*.sdf; do
      b=`basename $f .sdf`
      echo Processing ligand $b
      obabel -i sdf $f -o pdbqt -O $b.pdbqt -xh --partialcharge gasteiger
done
```

C.6.3: Convert Ligand SMILES into 3D structures

```
#! /bin/bash
# convert ligand SMILES into 3D strucutres with .pdbqt extension
for f in ./*.smi; do
      b=`basename $f .smi`
      echo Processing ligand $b
      obabel -i smi $f -o pdbqt -O $b.pdbqt -m -xh --gen3d --partialcharge
gasteiger
done
```

C.7: Frequency distribution of GPCRs 3D structures

```
# get data
GPCR_Pen <-read.csv("GPCR_Pen.txt",header = T, fill=TRUE, stringsAsFactors =
F)
```

```
GPCR_Pen <-as.data.frame(GPCR_Pen)
NROW(GPCR_Pen)

# unique gpcr on pdb
#rscbPDBgpcr <-read.csv("Finding_Unique_GPCR_PDB_ID_20220514.csv",header = T,
fill=T, sep = ",",
#                        stringsAsFactors = F, blank.lines.skip = F)
rscbPDBgpcr21 <- read.csv("Finding_Unique_GPCR_2022.csv",header = T, fill=T,
sep = ",",
                          stringsAsFactors = F, blank.lines.skip = F, skip =
1)
cond <- which(rscbPDBgpcr21[, 1] == "")
rscbPDBgpcr21 <- rscbPDBgpcr21[-cond,]
rscbPDBgpcr <- rscbPDBgpcr21
#cond <- which(rscbPDBgpcr[, 1] == "")
#rscbPDBgpcr <- rscbPDBgpcr[-cond,]

rscbPDBgpcr <-
rscbPDBgpcr[,c("PDB.ID","Deposition.Date","Release.Date","Accession.Code.s.")
]
colnames(rscbPDBgpcr) <-
c("PDB.ID","Deposition.Date","Release.Date","Uniprot.ID")
rscbPDBgpcr$Deposition.Date <- substring(rscbPDBgpcr$Deposition.Date, 1, 4)
rscbPDBgpcr$Release.Date <- substring(rscbPDBgpcr$Release.Date, 1, 4)

cond <- rscbPDBgpcr$Uniprot.ID %in% GPCR_Pen$ID
rscbPDBgpcr <- rscbPDBgpcr[cond,]

write.table(Uniprot_ID, file = "161 Unique GPCR.txt", sep = ",", row.names =
F, quote = F)
library(xlsx)
write.xlsx(PDB_ID_Date, file = "PDB_ID_Date.xlsx",
           sheetName = "Sheet1", col.names = TRUE, row.names = FALSE, append
= FALSE)

cond <- which(rscbPDBgpcr$Release.Date == "2021")
PDB_ID_Date <- rscbPDBgpcr [-cond,]

x <- barplot(table(rscbPDBgpcr), xaxt="n", col = "cyan", xlab = "Year", ylab
= "Frequency")
labs <- names(table(rscbPDBgpcr))
text(cex=1,x=x, y=-4.5, labs, xpd=T,srt=90)

library(lattice)
barchart(rscbPDBgpcr$Release.Date, ylab = "Frequency", xlab = "Year",
horizontal = F)
cumFreq <- ftable(rscbPDBgpcr$Release.Date)
l <- as.data.frame(cumFreq)
k <- cumsum(l$Freq)
l$Freq <- k
barchart(Freq ~ Var1, data = l, ylab = "Frequency", xlab = "Year", horizontal
= F)
```

APPENDIX D: MOTIF SEARCH

D.1: Cut Sequence into regions

```
library(seqinr)
ligID <- c("FQUAFMNPXPXOJE-UHFFFAOYSA-N","MLQFOEOUNIRULR-UHFFFAOYSA-
N","YKMSTUDOGGAEJH-UHFFFAOYSA-N",
           "USZPQRMQYJIDII-UHFFFAOYSA-N","BYBLEWFAAKGYCD-UHFFFAOYSA-
N","NKOPNLUYOHOGFZ-UHFFFAOYSA-N",
           "AJLFQFYMLRXVHV-UHFFFAOYSA-N","CLQVVBPDAXJGBV-UHFFFAOYSA-
N","DTZDSNQYNPNCPK-UHFFFAOYSA-N",
           "IKSHHOBCJKJKOG-UHFFFAOYSA-N","XLWJPQQFJNGUPA-UHFFFAOYSA-N")

num.ligID <- length(ligID)

for (k in 1:num.ligID) {
  # read the positions of the extracellular loops, helices, etc
  filName <- paste(ligID[k],"txt", sep = ".")
  positn <- read.table(file = filName, header = T, sep = "", stringsAsFactors
= F)
  # read the fasta file of the gpcrs
  fastaName <- paste(ligID[k], "fasta", sep = ".")
  fasta <- read.fasta(file = fastaName, seqtype = "AA", as.string = T,
whole.header = F,
                      strip.desc = F)
  # get the number of gpcr in fasta file
  num.seq <- (NROW(positn))/2

  for (j in 3:NCOL(positn)) {
    seq.fasta <- NULL
    seq.ID <- c()

    # row begin = i+(i-1), row end = i+((i+1)-1), i = 1:num.seq, rows =
1:NROW(positn)
    for (i in 1:num.seq) {
      strt <- i+(i-1)
      ends <- i+((i+1)-1)
      seq.sub <- substring(fasta[[i]][1], positn[strt, j], positn[ends, j])
      seq.ID[i] <- attr(fasta[[i]], "name")
      seq.list <- list(seq = seq.sub)
      seq.fasta <- rbind(seq.fasta, seq.list)
    }

    # write fasta file
    file.out <- paste(ligID[k], names(positn)[j], sep = "_")
    file.out <- paste(file.out, "txt", sep = ".")
    write.fasta(sequences = seq.fasta, names = seq.ID, file.out = file.out,
open = "w",
                nbchar = 60, as.string = T)
  }
}
rm(list = ls())
```

D.2: Cut Sequence into Extracellular loops

```
library(seqinr)
```

```
ligID <- c("FQUAFMNPXPXOJE-UHFFFAOYSA-N","MLQFOEOUNIRULR-UHFFFAOYSA-
N","YKMSTUDOGGAEJH-UHFFFAOYSA-N",
           "USZPQRMQYJIDII-UHFFFAOYSA-N","BYBLEWFAAKGYCD-UHFFFAOYSA-
N","NKOPNLUYOHOGFZ-UHFFFAOYSA-N",
           "AJLFQFYMLRXVHV-UHFFFAOYSA-N","CLQVVBPDAXJGBV-UHFFFAOYSA-
N","DTZDSNQYNPNCPK-UHFFFAOYSA-N",
           "IKSHHOBCJKJKOG-UHFFFAOYSA-N","XLWJPQQFJNGUPA-UHFFFAOYSA-N")

num.ligID <- length(ligID)

for (k in 1:num.ligID) {
  # read the positions of the extracellular loops, helices, etc
  filName <- paste(ligID[k],"txt", sep = ".")
  positn <- read.table(file = filName, header = T, sep = "", stringsAsFactors
= F)
  # read the fasta file of the gpcrs
  fastaName <- paste(ligID[k], "fasta", sep = ".")
  fasta <- read.fasta(file = fastaName, seqtype = "AA", as.string = T,
whole.header = F,
                      strip.desc = F)
  # get the number of gpcr in fasta file
  num.seq <- (NROW(positn))/2

  Eis <- c(3, 7, 11, 15) # column numbers of the Ei's in the positions of the
extracellular loops, helices, etc
  seq.fasta <- NULL
  seq.ID <- c()

  for (j in Eis) {

    # row begin = i+(i-1), row end = i+((i+1)-1), i = 1:num.seq, rows =
1:NROW(positn)
    for (i in 1:num.seq) {
      beg <- i+(i-1)
      end <- i+((i+1)-1)
      strt <- positn[beg, j]
      ends <- positn[end, j]

      if (j == 3){
        Positn_strt <- strt
      } else {
          Positn_strt <- strt - 5
      }
      Positn_ends <- ends + 5
      seq.sub <- substring(fasta[[i]][1], Positn_strt, Positn_ends)
      seq.InID <- paste(attr(fasta[[i]], "name"), names(positn)[j], sep = "")
      seq.ID <- c(seq.ID, seq.InID)
      seq.list <- list(seq = seq.sub)
      seq.fasta <- rbind(seq.fasta, seq.list)
    }
  }

  # write fasta file
  file.out <- paste(ligID[k], "E", sep = "_")
  file.out <- paste(file.out, "txt", sep = ".")
  write.fasta(sequences = seq.fasta, names = seq.ID, file.out = file.out,
open = "w",
```

```
              nbchar = 60, as.string = T)
}
rm(list = ls())
```

D.3: Cut Sequence into Modified Extracellular loops

```
library(seqinr)

ligID <- c("FQUAFMNPXPXOJE-UHFFFAOYSA-N","MLQFOEOUNIRULR-UHFFFAOYSA-
N","YKMSTUDOGGAEJH-UHFFFAOYSA-N",
           "USZPQRMQYJIDII-UHFFFAOYSA-N","BYBLEWFAAKGYCD-UHFFFAOYSA-
N","NKOPNLUYOHOGFZ-UHFFFAOYSA-N",
           "AJLFQFYMLRXVHV-UHFFFAOYSA-N","CLQVVBPDAXJGBV-UHFFFAOYSA-
N","DTZDSNQYNPNCPK-UHFFFAOYSA-N",
           "IKSHHOBCJKJKOG-UHFFFAOYSA-N","XLWJPQQFJNGUPA-UHFFFAOYSA-N")

num.ligID <- length(ligID)

for (k in 1:num.ligID) {
  # read the positions of the extracellular loops, helices, etc
  filName <- paste(ligID[k],"txt", sep = ".")
  positn <- read.table(file = filName, header = T, sep = "", stringsAsFactors
= F)
  # read the fasta file of the gpcrs
  fastaName <- paste(ligID[k], "fasta", sep = ".")
  fasta <- read.fasta(file = fastaName, seqtype = "AA", as.string = T,
whole.header = F,
                      strip.desc = F)
  # get the number of gpcr in fasta file
  num.seq <- (NROW(positn))/2

  Eis <- c(3, 7, 11, 15) # column numbers of the Ei's in the positions of the
extracellular loops, helices, etc

  for (j in Eis) {
    seq.fasta <- NULL
    seq.ID <- c()

    # row begin = i+(i-1), row end = i+((i+1)-1), i = 1:num.seq, rows =
1:NROW(positn)
    for (i in 1:num.seq) {
      beg <- i+(i-1)
      end <- i+((i+1)-1)
      strt <- positn[beg, j]
      ends <- positn[end, j]

      if (j == 3){
        Positn_strt <- strt
      } else {
          Positn_strt <- strt - 5
      }
      Positn_ends <- ends + 5
      seq.sub <- substring(fasta[[i]][1], Positn_strt, Positn_ends)
      seq.ID[i] <- attr(fasta[[i]], "name")
      seq.list <- list(seq = seq.sub)
      seq.fasta <- rbind(seq.fasta, seq.list)
    }
```

```
    # write fasta file
    file.out <- paste(ligID[k], names(positn)[j], sep = "_")
    file.out <- paste(file.out, "txt", sep = ".")
    write.fasta(sequences = seq.fasta, names = seq.ID, file.out = file.out,
open = "w",
                nbchar = 60, as.string = T)
  }
}
rm(list = ls())
```

D.4: Run MEME

```
#!/bin/bash

for dir in ./fasta_MEME_full/*/
do
      for file in $dir*.txt
      do
            meme $file -protein -oc ${file%.*} -nostatus -time 18000 -mod
zoops -nmotifs 10 -minw 3 -maxw 20
      done
done
```

APPENDIX E: ANALYSIS ON 3D STRUCTURES OF GPCRS

E.1: 3D Structural Comparisons

E.1.1: Running FATCAT

```
#! /bin/bash
# Run FATCAT
source ~/.bashrc
## source ~/.zshrc

declare -a list=("0HK" "7LD" "7MA" "8NU" "40F" "89F" "ADN" "GGL" "GLU" "SRO"
"Z99")
declare -a indx
declare -a num_prots
num_prots[0]="0HK_P08173_A_5dsg.pdb;0HK_P11229_A_5cxv.pdb"
num_prots[1]="7LD_P28223_A_6wgt.pdb;7LD_P41595_A_5tvn.pdb"
num_prots[2]="7MA_O43613_A_6tod.pdb;7MA_O43614_A_5wqc.pdb"
num_prots[3]="8NU_P14416_A_6cm4.pdb;8NU_P28223_A_6a93.pdb"
num_prots[4]="40F_Q14416_C_4xaq.pdb;40F_Q14832_C_4xar.pdb"
num_prots[5]="89F_P28222_A_5v54.pdb;89F_P28223_A_6wh4.pdb"
num_prots[6]="ADN_P29274_A_2ydo.pdb;ADN_P30542_A_6d9h_R.pdb"
num_prots[7]="GGL_O00222_C_6bsz.pdb;GGL_Q14416_C_5cni.pdb;GGL_Q14832_C_5cnk.p
db"
num_prots[8]="GLU_A0A173M0G2_TR2_5x2p_A.pdb;GLU_E9P5T5_NOT_4io2_A.pdb;GLU_P41
594_C_3lmk_A.pdb;GLU_P42264_NOT_3s9e_A.pdb;GLU_Q14416_C_7mtr_A.pdb"
num_prots[9]="SRO_P08908_A_7e2y_R.pdb;SRO_P37231_NOT_3adv_B.pdb"
```

```
num_prots[10]="Z99_P41594_C_7fd9_A.pdb;Z99_Q13255_C_3ks9_A.pdb;Z99_Q14831_C_3
mq4.pdb;Z99_Q14832_C_7wi6_A.pdb"

declare -i i=0
for dir in "${list[@]}"; do
    cd ./$dir

        #create an array out of the string separated by ;
        IFS=";" read -r -a arr <<< "${num_prots[$i]}"

        #get the length of the array arr
        len=`expr ${#arr[@]} - 1`

        #create all possible pairs of the indices of the protein and perform
APoc#
        set -- `seq 0 $len`
        for a; do
                shift
                for b; do
                        aa=`basename ${arr[$a]} .pdb`
                        bb=`basename ${arr[$b]} .pdb`
                        FATCAT -p1 ${arr[$a]} -p2 ${arr[$b]} -o
${aa}_${bb}_flex -m
                        FATCAT -p1 ${arr[$a]} -p2 ${arr[$b]} -o
${aa}_${bb}_rigid -m -r
                done
        done
    echo "Done with $dir"
    i=`expr $i + 1`
    cd ../
done
```

E.1.2: Extract Superimposed 3D Part

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  2 16:51:15 2022
This code maps the 3D structural alignment of two proteins to their
respective pdb files
and extract the portions of the 3D structures that were found to be similar.
The extracted portions are written to a pdb file.
This is done both for flexible and rigid 3D structural comparisons.
@author: kwabena
"""

import re
import csv
from itertools import combinations
from Bio import SeqIO

# set directory
import os
path =
'/Users/kwabena/Research/GPCR/Entire_work_organized/3D_Comparison_Pocket_over
lap_scores/3D_structural_comparison'
```

```
protein = [["0HK_P08173_A_5dsg","0HK_P11229_A_5cxv"],
["7LD_P28223_A_6wgt","7LD_P41595_A_5tvn"],
          ["7MA_O43613_A_6tod","7MA_O43614_A_5wqc"],
["8NU_P14416_A_6cm4","8NU_P28223_A_6a93"],
          ["40F_Q14416_C_4xaq","40F_Q14832_C_4xar"],
["89F_P28222_A_5v54","89F_P28223_A_6wh4"],

["ADN_P29274_A_2ydo","ADN_P30542_A_6d9h_R"],["GGL_O00222_C_6bsz","GGL_Q14416_
C_5cni","GGL_Q14832_C_5cnk"],

["GLU_A0A173M0G2_TR2_5x2p_A","GLU_E9P5T5_NOT_4io2_A","GLU_P41594_C_3lmk_A","G
LU_P42264_NOT_3s9e_A","GLU_Q14416_C_7mtr_A"],
          ["SRO_P08908_A_7e2y_R","SRO_P37231_NOT_3adv_B"],

["Z99_P41594_C_7fd9_A","Z99_Q13255_C_3ks9_A","Z99_Q14831_C_3mq4","Z99_Q14832_
C_7wi6_A"]]

folder = ["0HK", "7LD", "7MA", "8NU", "40F", "89F", "ADN", "GGL", "GLU",
"SRO", "Z99"]

columns = ['Ligand','Protein_Pair_1', 'Protein_Pair_2', '3D_RMSD_flex',
'3D_RMSD_rigid', 'Aligned_Seq_Identity_flex',
          'Aligned_Seq_Identity_rigid', 'Aligned_Seq_Similarity_flex',
'Aligned_Seq_Similarity_rigid']
results_file = '3D_Similar_RMSD.tsv'

# create a csv file with only column heads
os.chdir(path)
with open(results_file, "w") as outfile:
    writer = csv.writer(outfile, delimiter = "\t")
    writer.writerow(columns)
#outfile.close()

for indx, fold in enumerate(folder):
    path_list = [path, fold]
    paths = "/".join(path_list)
    os.chdir(paths)

    # create dict to hold the results
    all_columns = {}
    for i in range(len(columns)):
        all_columns.setdefault(columns[i],[])

    length = len(protein[indx])
    comb = combinations(range(0,length), 2)

    # loop over combinations of the pairs of protein pockets
    for j in comb:
        # Get the alignment file names and read them. Also read protein files

        flex_aln_name = "_".join([(protein[indx])[j[0]],
(protein[indx])[j[1]], 'flex.aln'])
        rigid_aln_name = "_".join([(protein[indx])[j[0]],
(protein[indx])[j[1]], 'rigid.aln'])

        flex_aln = open(flex_aln_name, "r")
        aln_similar_flex = flex_aln.readlines()
```

```
        rigid_aln = open(rigid_aln_name, "r")
        aln_similar_rigid = rigid_aln.readlines()

        prot_1 = [record.seq for record in
SeqIO.parse(".".join([(protein[indx])[j[0]], 'pdb']), "pdb-atom")]
        prot_1_seq = str(prot_1[0]).replace("X","")
        prot_2 = [record.seq for record in
SeqIO.parse(".".join([(protein[indx])[j[1]], 'pdb']), "pdb-atom")]
        prot_2_seq = str(prot_2[0]).replace("X","")

        prot_11 = open(".".join([(protein[indx])[j[0]], 'pdb']), "r")
        prot_11_pdb = prot_11.read()
        prot_22 = open(".".join([(protein[indx])[j[1]], 'pdb']), "r")
        prot_22_pdb = prot_22.read()

        # get the positions of the sequences in the pdb files
        regex = r"[\sA-Z]\d{1,}(?=\s{4})"
        position_prot_1 = re.finditer(regex, prot_11_pdb)
        position_prot_2 = re.finditer(regex, prot_22_pdb)

        position_prot_1_uniq = []
        for pos_prot_1 in position_prot_1:
            pos_pt_1 = pos_prot_1.group()
            pos_pt_1 = int(re.sub(r"\D","", pos_pt_1))
            if pos_pt_1 not in position_prot_1_uniq:
                position_prot_1_uniq.append(pos_pt_1)

        position_prot_2_uniq = []
        for pos_prot_2 in position_prot_2:
            pos_pt_2 = pos_prot_2.group()
            pos_pt_2 = int(re.sub(r"\D","", pos_pt_2))
            if pos_pt_2 not in position_prot_2_uniq:
                position_prot_2_uniq.append(pos_pt_2)

        # map the alignment to the pdb files and subset the pdb for the
blocks of the alignment
        position_chain_1_flex = []
        position_chain_2_flex = []
        position_chain_1_rigid = []
        position_chain_2_rigid = []

        prot_file_ = open(".".join([(protein[indx])[j[0]], 'pdb']), "r")
        prot_file_1 = prot_file_.readlines()
        prot_file__ = open(".".join([(protein[indx])[j[1]], 'pdb']), "r")
        prot_file_2 = prot_file__.readlines()

        # Flex
        temp_flex_1 = position_prot_1_uniq[:]
        temp_flex_2 = position_prot_2_uniq[:]
        for ii, line_aln_flex in enumerate(aln_similar_flex):

            if line_aln_flex[:7] == 'Chain 1':
                line_aln_flex = line_aln_flex.split(" ")
                ind_1 = position_prot_1_uniq.index(int(line_aln_flex[-2]))
#the index of the 1st AA in aln in position_prot_1_uniq
                indx__1 = [(i-14+ind_1) for i, ij in
enumerate(line_aln_flex[-1].replace("\n", "")) if ij=='-']
                if len(indx__1) > 0:
```

```
                    for ind in indx__1:
                            temp_flex_1.insert(ind, "-")

                block_indx_1 = [(i-14+ind_1) for i, ij in
enumerate(aln_similar_flex[(ii+1)].replace("\n", "")) if ij!=' ']
                if len(block_indx_1) > 0:
                    for jj in block_indx_1:
                        position_chain_1_flex.append(temp_flex_1[jj])

            if line_aln_flex[:7] == 'Chain 2':
                line_aln_flex = line_aln_flex.split(" ")
                ind_2 = position_prot_2_uniq.index(int(line_aln_flex[-2]))
#the index of the 1st AA in aln in position_prot_2_uniq
                indx__2 = [(ik-14+ind_2) for ik, ji in
enumerate(line_aln_flex[-1].replace("\n", "")) if ji=='-']
                if len(indx__2) > 0:
                    for ind in indx__2:
                        temp_flex_2.insert(ind, "-")

                block_indx_2 = [(ik-14+ind_2) for ik,ji in
enumerate(aln_similar_flex[(ii-1)].replace("\n", "")) if ji!=' ']
                if len(block_indx_2) > 0:
                    for ki in block_indx_2:
                        position_chain_2_flex.append(temp_flex_2[ki])

        #create file names
        flex_1 = "_".join([(protein[indx])[j[0]], 'flex_with',
(protein[indx])[j[1]]])
        flex_file_name_1 = ".".join([flex_1, "txt"])

        flex_2 = "_".join([(protein[indx])[j[1]], 'flex_with',
(protein[indx])[j[0]]])
        flex_file_name_2 = ".".join([flex_2, "txt"])

        #write 3D file of the alignment
        flex_file_1 = open(flex_file_name_1, "a+")  #protein 1 flex
        for indxx in position_chain_1_flex:
            if indxx != "-":
                for lines in prot_file_1[:-1]:
                    line = lines.split(" ")
                    lin = ' '.join(line).split()
                    len_lin = len(lin)
                    if lin[0] == "ATOM":
                        if len_lin == 12:
                            if lin[5] == str(indxx):
                                flex_file_1.write(lines)
                        if len_lin == 11:
                            lin_5 = re.sub(r"\D","", lin[4])
                            if lin_5 == str(indxx):
                                flex_file_1.write(lines)

        flex_file_1.write("TER") # Important for PDB file
        flex_file_1.close()

        flex_file_2 = open(flex_file_name_2, "a+")    #protein 2 flex
        for indxx in position_chain_2_flex:
            if indxx != "-":
                for lines in prot_file_2[:-1]:
```

```
                    line = lines.strip().split(" ")
                    lin = ' '.join(line).split()
                    len_lin = len(lin)
                    if lin[0] == "ATOM":
                        if len_lin == 12:
                            if lin[5] == str(indxx):
                                flex_file_2.write(lines)
                        elif len_lin == 11:
                            lin_5 = re.sub(r"\D","", lin[4])
                            if lin_5 == str(indxx):
                                flex_file_2.write(lines)

        flex_file_2.write("TER") # Important for PDB file
        flex_file_2.close()

        # Rigid
        temp_rigid_1 = position_prot_1_uniq[:]
        temp_rigid_2 = position_prot_2_uniq[:]
        for iii, line_aln_rigid in enumerate(aln_similar_rigid):

            if line_aln_rigid[:7] == 'Chain 1':
                line_aln_rigid = line_aln_rigid.split(" ")
                ind_1 = position_prot_1_uniq.index(int(line_aln_rigid[-2]))
#the index of the 1st AA in aln in position_prot_1_uniq
                indx__1 = [(i-14+ind_1) for i, ij in
enumerate(line_aln_rigid[-1].replace("\n", "")) if ij=='-']
                if len(indx__1) > 0:
                    for ind in indx__1:
                        temp_rigid_1.insert(ind, "-")

                block_indx_1 = [(jk-14+ind_1) for jk ,kj in
enumerate(aln_similar_rigid[(iii+1)].replace("\n", "")) if kj!=' ']
                if len(block_indx_1) > 0:
                    for jjj in block_indx_1:
                        position_chain_1_rigid.append(temp_rigid_1[jjj])

            if line_aln_rigid[:7] == 'Chain 2':
                line_aln_rigid = line_aln_rigid.split(" ")
                ind_2 = position_prot_2_uniq.index(int(line_aln_rigid[-2]))
#the index of the 1st AA in aln in position_prot_2_uniq
                indx__2 = [(i-14+ind_2) for i, ij in
enumerate(line_aln_rigid[-1].replace("\n", "")) if ij=='-']
                if len(indx__2) > 0:
                    for ind in indx__2:
                        temp_rigid_2.insert(ind, "-")

                block_indx_2 = [(kk-14+ind_2) for kk,ijk in
enumerate(aln_similar_rigid[(iii-1)].replace("\n", "")) if ijk!=' ']
                if len(block_indx_2) > 0:
                    for kkk in block_indx_2:
                        position_chain_2_rigid.append(temp_rigid_2[kkk])

        rigid_1 = "_".join([(protein[indx])[j[0]], 'rigid_with',
(protein[indx])[j[1]]])
        rigid_file_name_1 = ".".join([rigid_1, "txt"])

        rigid_2 = "_".join([(protein[indx])[j[1]], 'rigid_with',
(protein[indx])[j[0]]])
```

```
        rigid_file_name_2 = ".".join([rigid_2, "txt"])

        #write 3D file of the alignment
        rigid_file_1 = open(rigid_file_name_1, "a+")  #protein 1 rigid
        for indxx in position_chain_1_rigid:
            if indxx != "-":
                for lines in prot_file_1[:-1]:
                    line = lines.strip().split(" ")
                    lin = ' '.join(line).split()
                    len_lin = len(lin)
                    if lin[0] == "ATOM":
                        if len_lin == 12:
                            if lin[5] == str(indxx):
                                rigid_file_1.write(lines)
                        elif len_lin == 11:
                            lin_5 = re.sub(r"\D","", lin[4])
                            if lin_5 == str(indxx):
                                rigid_file_1.write(lines)

        rigid_file_1.write("TER") # Important for PDB file
        rigid_file_1.close()

        rigid_file_2 = open(rigid_file_name_2, "a+")    #protein 2 rigid
        for indxx in position_chain_2_flex:
            if indxx != "-":
                for lines in prot_file_2[:-1]:
                    line = lines.strip().split(" ")
                    lin = ' '.join(line).split()
                    len_lin = len(lin)
                    if lin[0] == "ATOM":
                        if len_lin == 12:
                            if lin[5] == str(indxx):
                                rigid_file_2.write(lines)
                        elif len_lin == 11:
                            lin_5 = re.sub(r"\D","", lin[4])
                            if lin_5 == str(indxx):
                                rigid_file_2.write(lines)

        rigid_file_2.write("TER") # Important for PDB file
        rigid_file_2.close()


        # save results to dict created above
        all_columns['Ligand'].append(fold)
        all_columns['Protein_Pair_1'].append(".".join([(protein[indx])[j[0]],
'pdb']))
        all_columns['Protein_Pair_2'].append(".".join([(protein[indx])[j[1]],
'pdb']))
        all_columns['3D_RMSD_flex'].append(aln_similar_flex[1].split()[9])
        all_columns['3D_RMSD_rigid'].append(aln_similar_rigid[1].split()[9])

all_columns['Aligned_Seq_Identity_flex'].append(aln_similar_flex[2].split()[5
].replace("%",""))

all_columns['Aligned_Seq_Identity_rigid'].append(aln_similar_rigid[2].split()
[5].replace("%",""))
```

```
all_columns['Aligned_Seq_Similarity_flex'].append(aln_similar_flex[2].split()
[7].replace("%","").replace("\n",""))

all_columns['Aligned_Seq_Similarity_rigid'].append(aln_similar_rigid[2].split
()[7].replace("%","").replace("\n",""))

    # change directory and write the RMSD results
    os.chdir(path)
    with open(results_file, "a+") as outfiles:
        writer = csv.writer(outfiles, delimiter = "\t")
        writer.writerows(zip(*[all_columns[key] for key in columns]))

outfiles.close()
```

E.2: Binding Pocket Prediction and Comparisons

E.2.1: Binding Pocket Prediction

E.2.1.1: Clean 3D Structures

```
#! /bin/bash
## this script removes unwanted lines in the pdb files to put them in the
right
## input format for P2Rank

declare -a list=("0HK" "7LD" "7MA" "8NU" "40F" "89F" "97V" "ADN" "GGL" "GLU"
"SRO" "Z99")

for dir in "${list[@]}"; do
    cd ./$dir

    ## convert pdb files to txt files
    for f in ./*.pdb; do
        mv "$f" "${f%.*}.txt"
    done

    for file in ./*.txt; do
        echo $file
        b=`basename $file .txt`
        grep ATOM $file > ${b}.pdb
        obabel -i pdb ${b}.pdb -o pdbqt -O ${b}.pdbqt -xh --partialcharge
gasteiger
    done
    rm *.txt

    cd ../
done
```

E.2.1.2: Run P2Rank

```
#! /bin/bash
# Run P2Rank

declare -a list=("0HK" "7LD" "7MA" "8NU" "40F" "89F" "97V" "ADN" "GGL" "GLU"
"SRO" "Z99")
declare -a indx
```

```
declare -a num_prots
num_prots[0]="0HK_P08173_A_5dsg.pdb;0HK_P11229_A_5cxv.pdb"
num_prots[1]="7LD_P28223_A_6wgt.pdb;7LD_P41595_A_5tvn.pdb"
num_prots[2]="7MA_O43613_A_6tod.pdb;7MA_O43614_A_5wqc.pdb"
num_prots[3]="8NU_P14416_A_6cm4.pdb;8NU_P28223_A_6a93.pdb"
num_prots[4]="40F_Q14416_C_4xaq.pdb;40F_Q14832_C_4xar.pdb"
num_prots[5]="89F_P28222_A_5v54.pdb;89F_P28223_A_6wh4.pdb"
num_prots[6]="97V_P43220_B_5vex.pdb;97V_P47871_B_5xf1.pdb"
num_prots[7]="ADN_P29274_A_2ydo.pdb;ADN_P30542_A_6d9h_R.pdb"
num_prots[8]="GGL_O00222_C_6bsz.pdb;GGL_Q14416_C_5cni.pdb;GGL_Q14832_C_5cnk.p
db"
num_prots[9]="GLU_A0A173M0G2_TR2_5x2p_A.pdb;GLU_E9P5T5_NOT_4io2_A.pdb;GLU_P41
594_C_3lmk_A.pdb;GLU_P42264_NOT_3s9e_A.pdb;GLU_Q14416_C_7mtr_A.pdb"
num_prots[10]="SRO_P08908_A_7e2y_R.pdb;SRO_P37231_NOT_3adv_B.pdb"
num_prots[11]="Z99_P41594_C_7fd9_A.pdb;Z99_Q13255_C_3ks9_A.pdb;Z99_Q14831_C_3
mq4.pdb;Z99_Q14832_C_7wi6_A.pdb"

declare -a prot_indx
declare -i i=0
declare -i j
for dir in "${list[@]}"; do
    cd ./$dir

        IFS=";" read -r -a arr <<< "${num_prots[$i]}"

        #get the length of the array arr
        len=`expr ${#arr[@]} - 1`

        #create all possible pairs of the indices of the protein and perform
P2Rank
        set -- `seq 0 $len`
        for a; do
            prank predict -o ./ -f ./${arr[$a]} -threads 8
        done
    i=`expr $i + 1`
    cd ../
done
```

E.2.1.3: Get 3D Structures of Predicted Pockets

```
###
# This code creates a PDB file of the pockets with .txt extension
# it also create configuration files for AutoDock vina
###

# set directory
import os
path = '/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control/Binding_pocket_prediction_and_comparison/Pocket
_Predictions'
os.chdir(path)
import csv

protein = [["0HK_P08173_A_5dsg.pdb","0HK_P11229_A_5cxv.pdb"],
["7LD_P28223_A_6wgt.pdb","7LD_P41595_A_5tvn.pdb"],
["7MA_O43613_A_6tod.pdb","7MA_O43614_A_5wqc.pdb"],["8NU_P14416_A_6cm4.pdb","8
NU_P28223_A_6a93.pdb"],["40F_Q14416_C_4xaq.pdb","40F_Q14832_C_4xar.pdb"],["89
F_P28222_A_5v54.pdb","89F_P28223_A_6wh4.pdb"],["ADN_P29274_A_2ydo.pdb","ADN_P
```

```
30542_A_6d9h_R.pdb"],["GGL_O00222_C_6bsz.pdb","GGL_Q14416_C_5cni.pdb","GGL_Q1
4832_C_5cnk.pdb"],["GLU_A0A173M0G2_TR2_5x2p_A.pdb","GLU_E9P5T5_NOT_4io2_A.pdb
","GLU_P41594_C_3lmk_A.pdb","GLU_P42264_NOT_3s9e_A.pdb","GLU_Q14416_C_7mtr_A.
pdb"],["SRO_P08908_A_7e2y_R.pdb","SRO_P37231_NOT_3adv_B.pdb"],["Z99_P41594_C_
7fd9_A.pdb","Z99_Q13255_C_3ks9_A.pdb","Z99_Q14831_C_3mq4.pdb","Z99_Q14832_C_7
wi6_A.pdb"]]

pkt_row_num = [[0,0], [0,0], [0,0], [0,0], [0,0], [0,0], [0,0], [0,0,1],
[0,0,0,0,1], [0,0], [0,0,4,9]]
folder = ["0HK", "7LD", "7MA", "8NU", "40F", "89F", "ADN", "GGL", "GLU",
"SRO", "Z99"]

for ii, fold in enumerate(folder):
    path_list = [path, fold]
    paths = "/".join(path_list)
    os.chdir(paths)
    for jj, prot in enumerate(protein[ii]):
        file_name = "_".join([prot, "predictions.csv"])
        file = open(file_name)
        csv_file = file.readlines()[1:]
        pkt_info = csv_file[pkt_row_num[ii][jj]].split(",")

        #write docking configuration file
        center_x = pkt_info[6].strip()
        center_y = pkt_info[7].strip()
        center_z = pkt_info[8].strip()
        receptor_name = "".join([prot, "qt"])
        config_name = "_".join([prot.split(".")[0], "config.txt"])
        #write docking configuration file content and close file
        config_file = open(config_name, "w")
        config_file.write("receptor = %s\n\n" %receptor_name)
        config_file.write("center_x = %s\n" %center_x)
        config_file.write("center_y = %s\n" %center_y)
        config_file.write("center_z = %s\n\n" %center_z)
        config_file.write("size_x = 20\n")
        config_file.write("size_y = 20\n")
        config_file.write("size_z = 20\n\n")
        config_file.write("num_modes = 1\n")
        config_file.write("exhaustiveness = 9")
        config_file.close()

        #get pockets AA and write a txt file of the atoms coordinates
        split_val = pkt_info[9].strip()[0:2]  # chain name eg A, B, C with _
eg A_, B_, C_
        AA_ind = pkt_info[9].split(split_val)[1:] # get the positions of the
amino acids in pocket
        AA_ind = list(map(int, AA_ind))
        AA_ind.sort()
        AA_indx = list(map(str, AA_ind))
        prot_file_pdb = open(prot, "r")
        prot_file = prot_file_pdb.readlines()
        #create pocket file name
        pkt_num = pkt_row_num[ii][jj] + 1
        pkt_name = "_".join([prot.split(".")[0], "pkt", str(pkt_num)])
        pkt_file_name = ".".join([pkt_name, "txt"])
        #write pocket file
        pkt_file = open(pkt_file_name, "a+")
```

```
        pkt_file.write("PKT        11    101    " + pkt_name + "\n") #
Important for APoc
        for indx in AA_indx:
            for lines in prot_file[:-1]:
                line = lines.split(" ")
                lin = ' '.join(line).split()
                if lin[0] == "ATOM":
                    lin_5 = lin[5].split(split_val[0])
                    lin_5 = ' '.join(lin_5).split()[0]
                    if lin_5 == indx:
                        pkt_file.write(lines)
        pkt_file.write("TER") # Important for PDB file
        pkt_file.close()
        prot_file_pdb.close()
```

E.2.2: Binding Pocket Comparison

E.2.2.1: Append Pocket 3D files to their Protein 3D file

```
###
# This python file is used to add the pocket file to the GPCR 3D structure
file
###

# set directory
import os
path = '/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control/Binding_pocket_prediction_and_comparison/APoc'
os.chdir(path)

protein = [["0HK_P08173_A_5dsg.pdb","0HK_P11229_A_5cxv.pdb"],
["7LD_P28223_A_6wgt.pdb","7LD_P41595_A_5tvn.pdb"],
["7MA_O43613_A_6tod.pdb","7MA_O43614_A_5wqc.pdb"],["8NU_P14416_A_6cm4.pdb","8
NU_P28223_A_6a93.pdb"],["40F_Q14416_C_4xaq.pdb","40F_Q14832_C_4xar.pdb"],["89
F_P28222_A_5v54.pdb","89F_P28223_A_6wh4.pdb"],["ADN_P29274_A_2ydo.pdb","ADN_P
30542_A_6d9h_R.pdb"],["GGL_O00222_C_6bsz.pdb","GGL_Q14416_C_5cni.pdb","GGL_Q1
4832_C_5cnk.pdb"],["GLU_A0A173M0G2_TR2_5x2p_A.pdb","GLU_E9P5T5_NOT_4io2_A.pdb
","GLU_P41594_C_3lmk_A.pdb","GLU_P42264_NOT_3s9e_A.pdb","GLU_Q14416_C_7mtr_A.
pdb"],["SRO_P08908_A_7e2y_R.pdb","SRO_P37231_NOT_3adv_B.pdb"],["Z99_P41594_C_
7fd9_A.pdb","Z99_Q13255_C_3ks9_A.pdb","Z99_Q14831_C_3mq4.pdb","Z99_Q14832_C_7
wi6_A.pdb"]]

pkt_row_num = [[0,0], [0,0], [0,0], [0,0], [0,0], [0,0], [0,0], [0,0,1],
[0,0,0,0,1], [0,0], [0,0,4,9]]
folder = ["0HK", "7LD", "7MA", "8NU", "40F", "89F", "ADN", "GGL", "GLU",
"SRO", "Z99"]

for ii, fold in enumerate(folder):
    path_list = [path, fold]
    paths = "/".join(path_list)
    os.chdir(paths)
    for jj, prot in enumerate(protein[ii]):
        file = open(prot, "a+")

        #
        pkt_num = pkt_row_num[ii][jj] + 1
        pkt_name = "_".join([prot.split(".")[0], "pkt", str(pkt_num)])
```

```
        pkt_file_name = ".".join([pkt_name, "txt"])
        pkt_file = open(pkt_file_name, "r")
        pkt = pkt_file.read()
        file.write(pkt)

        pkt_file.close()
        file.close()
```

E.2.2.2: Run APoc

```
#! /bin/bash
# Run APoc

declare -a list=("0HK" "7LD" "7MA" "8NU" "40F" "89F" "ADN" "GGL" "GLU" "SRO"
"Z99")
declare -a indx
declare -a num_prots
num_prots[0]="0HK_P08173_A_5dsg.pdb;0HK_P11229_A_5cxv.pdb"
num_prots[1]="7LD_P28223_A_6wgt.pdb;7LD_P41595_A_5tvn.pdb"
num_prots[2]="7MA_O43613_A_6tod.pdb;7MA_O43614_A_5wqc.pdb"
num_prots[3]="8NU_P14416_A_6cm4.pdb;8NU_P28223_A_6a93.pdb"
num_prots[4]="40F_Q14416_C_4xaq.pdb;40F_Q14832_C_4xar.pdb"
num_prots[5]="89F_P28222_A_5v54.pdb;89F_P28223_A_6wh4.pdb"
num_prots[6]="ADN_P29274_A_2ydo.pdb;ADN_P30542_A_6d9h_R.pdb"
num_prots[7]="GGL_O00222_C_6bsz.pdb;GGL_Q14416_C_5cni.pdb;GGL_Q14832_C_5cnk.p
db"
num_prots[8]="GLU_A0A173M0G2_TR2_5x2p_A.pdb;GLU_E9P5T5_NOT_4io2_A.pdb;GLU_P41
594_C_3lmk_A.pdb;GLU_P42264_NOT_3s9e_A.pdb;GLU_Q14416_C_7mtr_A.pdb"
num_prots[9]="SRO_P08908_A_7e2y_R.pdb;SRO_P37231_NOT_3adv_B.pdb"
num_prots[10]="Z99_P41594_C_7fd9_A.pdb;Z99_Q13255_C_3ks9_A.pdb;Z99_Q14831_C_3
mq4.pdb;Z99_Q14832_C_7wi6_A.pdb"

declare -i i=0
for dir in "${list[@]}"; do
    cd ./$dir

        #create an array out of the string separated by ;
        IFS=";" read -r -a arr <<< "${num_prots[$i]}"

        #get the length of the array arr
        len=`expr ${#arr[@]} - 1`

        #create all possible pairs of the indices of the protein and perform
APoc
        set -- `seq 0 $len`
        for a; do
                shift
                for b; do
                        /Users/kwabena/apoc/bin/apoc ${arr[$a]} ${arr[$b]} -
pvol 50 -plen 5 > ${arr[$a]}_vs_${arr[$b]}_pocket_compare_results.txt
                done
        done
    i=`expr $i + 1`
    cd ../
done
```

E.2.2.3: Parse and Gather Pocket Comparison Results

```
###
# This python file is used to parese APoc bindiing pockets comparison results
text
# files and write them to a csv file
# This is done for all ligands at a time
###

from itertools import combinations
import csv
# set directory
import os
path = '/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control/Binding_pocket_prediction_and_comparison/APoc'
os.chdir(path)

protein = [["0HK_P08173_A_5dsg.pdb","0HK_P11229_A_5cxv.pdb"],
["7LD_P28223_A_6wgt.pdb","7LD_P41595_A_5tvn.pdb"],
["7MA_O43613_A_6tod.pdb","7MA_O43614_A_5wqc.pdb"],["8NU_P14416_A_6cm4.pdb","8
NU_P28223_A_6a93.pdb"],["40F_Q14416_C_4xaq.pdb","40F_Q14832_C_4xar.pdb"],["89
F_P28222_A_5v54.pdb","89F_P28223_A_6wh4.pdb"],["ADN_P29274_A_2ydo.pdb","ADN_P
30542_A_6d9h_R.pdb"],["GGL_O00222_C_6bsz.pdb","GGL_Q14416_C_5cni.pdb","GGL_Q1
4832_C_5cnk.pdb"],["GLU_A0A173M0G2_TR2_5x2p_A.pdb","GLU_E9P5T5_NOT_4io2_A.pdb
","GLU_P41594_C_3lmk_A.pdb","GLU_P42264_NOT_3s9e_A.pdb","GLU_Q14416_C_7mtr_A.
pdb"],["SRO_P08908_A_7e2y_R.pdb","SRO_P37231_NOT_3adv_B.pdb"],["Z99_P41594_C_
7fd9_A.pdb","Z99_Q13255_C_3ks9_A.pdb","Z99_Q14831_C_3mq4.pdb","Z99_Q14832_C_7
wi6_A.pdb"]]

folder = ["0HK", "7LD", "7MA", "8NU", "40F", "89F", "ADN", "GGL", "GLU",
"SRO", "Z99"]

columns = ['Ligand_ID','pocket_1', 'pocket_2', 'PS_score', 'P_value']
out_results_file_name = 'Combine_pocket_comp_results.tsv'

# create a csv file with only column heads
with open(out_results_file_name, "w") as outfile:
    writer = csv.writer(outfile, delimiter = "\t")
    writer.writerow(columns)

    all_columns = {}
    for i in range(len(columns)):
        all_columns.setdefault(columns[i],[])

    for indx, fold in enumerate(folder):
        path_list = [path, fold]
        paths = "/".join(path_list)
        os.chdir(paths)

        length = len(protein[indx])
        combs = combinations(range(0,length), 2)
        for j in list(combs):
            # Get the file names and read them j = (0, 1)
            results_file_name = "_".join([(protein[indx])[j[0]], 'vs',
(protein[indx])[j[1]], 'pocket_compare_results.txt'])
            lines = []
            with open(results_file_name, "r") as Results:
                for line in Results:
                    if not line.isspace():
                        line = line.strip()
```

```
                    lines.append(line)
            Results.close()

            # parse the text file and pick info needed
            for num, line in enumerate(lines,1):
                column = line.split()
                if column[0] == ">>>>>>>>>>>>>>>>>>>>>>>>>":
                    if column[1] == "Pocket":
                        if lines[num] == "The number of values to be sorted
is not positive.":
                            continue
                        all_columns['Ligand_ID'].append(fold)

all_columns['pocket_1'].append(lines[num].split()[7].split(':')[1])

all_columns['pocket_2'].append(lines[1+num].split()[7].split(':')[1])
                        vals_1 = lines[2+num].split(',')

all_columns['PS_score'].append(vals_1[0].split('=')[1])

all_columns['P_value'].append(vals_1[1].split('=')[1])

    #append the results to the tsv file
    writer.writerows(zip(*[all_columns[key] for key in columns]))
outfile.close()
```

E.3: Pockets and 3D Superimposed Part Overlap Scores

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Feb  4 20:52:19 2022
This code scores the overlap between the pockets of protein A and the part
of the protein A which is similar to protein B. Same is done for protein B.
This process is done both for flexible comaprison and rigid comparison of
protein A and protein B.
The code loop over each of the ligand folders, and the proteins in them and
the number of pockets for each protein.
Finally, it writes all the results as one file.
@author: Kwabena Owusu Dankwah
"""

from itertools import permutations
import os
import re
import csv
path = '/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control/3D_Comparison_Pocket_overlap_scores/Overlap_sco
res'
os.chdir(path)

protein = [["0HK_P08173_A_5dsg","0HK_P11229_A_5cxv"],
["7LD_P28223_A_6wgt","7LD_P41595_A_5tvn"],
          ["7MA_O43613_A_6tod","7MA_O43614_A_5wqc"],
["8NU_P14416_A_6cm4","8NU_P28223_A_6a93"],
          ["40F_Q14416_C_4xaq","40F_Q14832_C_4xar"],
["89F_P28222_A_5v54","89F_P28223_A_6wh4"],
```

```
["ADN_P29274_A_2ydo","ADN_P30542_A_6d9h_R"],["GGL_O00222_C_6bsz","GGL_Q14416_
C_5cni","GGL_Q14832_C_5cnk"],

["GLU_A0A173M0G2_TR2_5x2p_A","GLU_E9P5T5_NOT_4io2_A","GLU_P41594_C_3lmk_A","G
LU_P42264_NOT_3s9e_A","GLU_Q14416_C_7mtr_A"],
          ["SRO_P08908_A_7e2y_R","SRO_P37231_NOT_3adv_B"],

["Z99_P41594_C_7fd9_A","Z99_Q13255_C_3ks9_A","Z99_Q14831_C_3mq4","Z99_Q14832_
C_7wi6_A"]]

folder = ["0HK", "7LD", "7MA", "8NU", "40F", "89F", "ADN", "GGL", "GLU",
"SRO", "Z99"]

pkt_num =
[[1,1],[1,1],[1,1],[1,1],[1,1],[1,1],[1,1],[1,2,1],[1,1,1,1,2],[1,1],[1,5,1,1
0]]

columns = ['ligand_ID', 'Proteins_Compared', 'Pocket',
'P3D_similar_Coincide_pkt_flex', 'P3D_similar_Coincide_pkt_rigid']

def intersection_len(list1, list2):
    list3 = [value for value in list1 if value in list2]
    return len(list3)

def score_overlap(position_pocket, position_p3d_similar):
    len_pos_pkt = len(position_pocket)
    len_intersect = intersection_len(position_pocket, position_p3d_similar)
    score = len_intersect/len_pos_pkt
    return score

results_file = '3D_Similar_Overlap_Pkt.tsv'
# create a csv file with only column heads
with open(results_file, "w") as outfile:
    writer = csv.writer(outfile, delimiter = "\t")
    writer.writerow(columns)

# loop over the ligand folders
for indx, fold in enumerate(folder):
    path_list = [path, fold]
    paths = "/".join(path_list)
    os.chdir(paths)

    # create dict to hold the results
    all_columns = {}
    for i in range(len(columns)):
        all_columns.setdefault(columns[i],[])

    length = len(protein[indx])
    perm = permutations(range(0,length), 2)

    # loop over permutations of the pairs of protein pockets
    for j in list(perm):
        # Get the file names and read them
        proteins_comped = "_".join([(protein[indx])[j[0]], 'with',
(protein[indx])[j[1]]])
```

```
        flex = "_".join([(protein[indx])[j[0]], 'flex_with',
(protein[indx])[j[1]]])
        rigid = "_".join([(protein[indx])[j[0]], 'rigid_with',
(protein[indx])[j[1]]])
        flex_file_name = ".".join([flex, "txt"])
        rigid_file_name = ".".join([rigid, "txt"])

        flex_file = open(flex_file_name)
        d_similar_flex = flex_file.read()

        rigid_file = open(rigid_file_name)
        d_similar_rigid = rigid_file.read()

        # get the unique positions
        regex = r"[\sA-Z]\d{1,}(?=\s{4})"
        position_flex = re.finditer(regex, d_similar_flex)
        position_rigid = re.finditer(regex, d_similar_rigid)

        position_flex_uniq = []
        for pos_flex in position_flex:
            pos_fl = pos_flex.group()
            pos_fl = int(re.sub(r"\D","", pos_fl))
            if pos_fl not in position_flex_uniq:
                position_flex_uniq.append(pos_fl)

        position_rigid_uniq = []
        for pos_rigid in position_rigid:
            pos_ri = pos_rigid.group()
            pos_ri = int(re.sub(r"\D","", pos_ri))
            if pos_ri not in position_rigid_uniq:
                position_rigid_uniq.append(pos_ri)

        # compare pockets to the portions of the proteins that are 3D
structurally similar
        pkt_name = "_".join([(protein[indx])[j[0]], 'pkt',
str((pkt_num[indx])[j[0]])])
        pkt_file_name = ".".join([pkt_name, 'txt'])
        pkt_file = open(pkt_file_name)
        pkt = pkt_file.read()

        position_pkt = re.finditer(regex, pkt)
        position_pkt_uniq = []
        for pos_pkt in list(position_pkt)[2:]:
            pos_pk = pos_pkt.group()
            pos_pk = int(re.sub(r"\D","", pos_pk))
            if pos_pk not in position_pkt_uniq:
                position_pkt_uniq.append(pos_pk)

        # save results to dict created above
        all_columns['ligand_ID'].append(fold)
        all_columns['Proteins_Compared'].append(proteins_comped)
        all_columns['Pocket'].append(pkt_name)

all_columns['P3D_similar_Coincide_pkt_flex'].append(score_overlap(position_pk
t_uniq, position_flex_uniq))

all_columns['P3D_similar_Coincide_pkt_rigid'].append(score_overlap(position_p
kt_uniq, position_rigid_uniq))
```

```
    # change directory and write the results
    os.chdir(path)
    with open(results_file, "a+") as outfiles:
        writer = csv.writer(outfiles, delimiter = "\t")
        writer.writerows(zip(*[all_columns[key] for key in columns]))
outfiles.close()
```

APPENDIX F: GPCR LIGAND DOCKING AND LIGAND POSE AND CONFORMATION

F.1: Run AutoDock Vina

```
#! /bin/bash
# do not delete the line above

# this code performs docking between one ligand and multiple proteins
declare -a list=("0HK" "7LD" "7MA" "8NU" "40F" "89F" "ADN" "GGL" "GLU" "SRO"
"Z99")

declare -a num_prots
num_prots[0]="0HK_P08173_A_5dsg.pdbqt;0HK_P11229_A_5cxv.pdbqt"
num_prots[1]="7LD_P28223_A_6wgt.pdbqt;7LD_P41595_A_5tvn.pdbqt"
num_prots[2]="7MA_O43613_A_6tod.pdbqt;7MA_O43614_A_5wqc.pdbqt"
num_prots[3]="8NU_P14416_A_6cm4.pdbqt;8NU_P28223_A_6a93.pdbqt"
num_prots[4]="40F_Q14416_C_4xaq.pdbqt;40F_Q14832_C_4xar.pdbqt"
num_prots[5]="89F_P28222_A_5v54.pdbqt;89F_P28223_A_6wh4.pdbqt"
num_prots[6]="ADN_P29274_A_2ydo.pdbqt;ADN_P30542_A_6d9h_R.pdbqt"
num_prots[7]="GGL_O00222_C_6bsz.pdbqt;GGL_Q14416_C_5cni.pdbqt;GGL_Q14832_C_5c
nk.pdbqt"
num_prots[8]="GLU_A0A173M0G2_TR2_5x2p_A.pdbqt;GLU_E9P5T5_NOT_4io2_A.pdbqt;GLU
_P41594_C_3lmk_A.pdbqt;GLU_P42264_NOT_3s9e_A.pdbqt;GLU_Q14416_C_7mtr_A.pdbqt"
num_prots[9]="SRO_P08908_A_7e2y_R.pdbqt;SRO_P37231_NOT_3adv_B.pdbqt"
num_prots[10]="Z99_P41594_C_7fd9_A.pdbqt;Z99_Q13255_C_3ks9_A.pdbqt;Z99_Q14831
_C_3mq4.pdbqt;Z99_Q14832_C_7wi6_A.pdbqt"

i=0
for dir in "${list[@]}"; do
    cd ./$dir
    IFS=";" read -r -a arr <<< "${num_prots[$i]}"
    for prot in "${arr[@]}"; do  # in the ligand folder of ligands that bind
to the protein
        b=`basename $prot .pdbqt`      # separate the name from the file
extension
        echo Processing Protein $b and $dir  # print which protein is being
worked on

        vina --config ${b}_config.txt --ligand
../Control_Data_Ligands/${dir}.pdbqt --out ${b}_out.pdbqt --log ${b}_log.txt

    done
    ((i=$i+1))
    cd ../
done
```

F.2: Parse and Gather Binding Affinities from Docking

```
###
# This python file is used to parese APoc bindiing pockets comparison results
text
# files and write them to a csv file
# This is done for a single ligand at a time
###

import csv
# set directory
import os
path = '/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control/AutoDock_ligands_Proteins'
os.chdir(path)

input_file = [["0HK_P08173_A_5dsg","0HK_P11229_A_5cxv"],
["7LD_P28223_A_6wgt","7LD_P41595_A_5tvn"],
["7MA_O43613_A_6tod","7MA_O43614_A_5wqc"],["8NU_P14416_A_6cm4","8NU_P28223_A_
6a93"],["40F_Q14416_C_4xaq","40F_Q14832_C_4xar"],["89F_P28222_A_5v54","89F_P2
8223_A_6wh4"],["ADN_P29274_A_2ydo","ADN_P30542_A_6d9h_R"],["GGL_O00222_C_6bsz
","GGL_Q14416_C_5cni","GGL_Q14832_C_5cnk"],["GLU_A0A173M0G2_TR2_5x2p_A","GLU_
E9P5T5_NOT_4io2_A","GLU_P41594_C_3lmk_A","GLU_P42264_NOT_3s9e_A","GLU_Q14416_
C_7mtr_A"],["SRO_P08908_A_7e2y_R","SRO_P37231_NOT_3adv_B"],["Z99_P41594_C_7fd
9_A","Z99_Q13255_C_3ks9_A","Z99_Q14831_C_3mq4","Z99_Q14832_C_7wi6_A"]]

ligand = ["0HK", "7LD", "7MA", "8NU", "40F", "89F", "ADN", "GGL", "GLU",
"SRO", "Z99"]

# define the APoc results text files to parse and the csv file to write to
results_file = 'AutoDock_vina_Results.tsv'
#counts = [[1,1], [1,1], [1,1], [1,1], [1,1], [1,1], [1,1], [1,1,1],
[1,1,1,1,1], [1,1], [1,1,1,1]]

# function to parse results text file
columns = ['Ligand_ID', 'Protein_ID', 'Affinity_kcal_mol']


# create a csv file with only column heads
with open(results_file, "w") as outfile:
    writer = csv.writer(outfile, delimiter = "\t")
    writer.writerow(columns)

    all_columns = {}
    for i in range(len(columns)):
        all_columns.setdefault(columns[i],[])

    for ii, file in enumerate(input_file):

        paths = "/".join([path, str(ligand[ii])])
        os.chdir(paths)

        # read text file
        for log_file in file:

            file_name = '.'.join(['_'.join([log_file, 'log']),'txt'])

            with open(file_name, "r") as Results:
```

```
                for line in Results:
                    if not line.isspace():
                        line = line.strip()
                        column = line.split()
                        if column[0] == "1":
                            all_columns['Ligand_ID'].append(ligand[ii])
                            all_columns['Protein_ID'].append(log_file)

all_columns['Affinity_kcal_mol'].append(column[1])
            Results.close()

    # append the results to the csv file
    # zip() transpose the original data
    writer.writerows(zip(*[all_columns[key] for key in columns]))
outfile.close()
```

F.3: Docked Ligand Conformation

```
###
# This python file is used to align one set of ligands with another set
# in a pairwise manner
# The RMSD from the alignment are written to a csv file
# This is done for a single ligand at a time
###

import pymol
from pymol import cmd
# set directory
import os
os.chdir('/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control/Ligand_Pose_Comformation_Docked')
import csv
from itertools import combinations

ligs = ["0HK","7LD","7MA","8NU","40F","89F","ADN","GGL","GLU","SRO","Z99"]
protein = [["P08173_A_5dsg","P11229_A_5cxv"],
["P28223_A_6wgt","P41595_A_5tvn"],
["O43613_A_6tod","O43614_A_5wqc"],["P14416_A_6cm4","P28223_A_6a93"],["Q14416_
C_4xaq","Q14832_C_4xar"],["P28222_A_5v54","P28223_A_6wh4"],["P29274_A_2ydo","
P30542_A_6d9h_R"],
["O00222_C_6bsz","Q14416_C_5cni","Q14832_C_5cnk"],["A0A173M0G2_TR2_5x2p_A","E
9P5T5_NOT_4io2_A","P41594_C_3lmk_A","P42264_NOT_3s9e_A","Q14416_C_7mtr_A"],["
P08908_A_7e2y_R","P37231_NOT_3adv_B"],["P41594_C_7fd9_A","Q13255_C_3ks9_A","Q
14831_C_3mq4","Q14832_C_7wi6_A"]]

pkt_counts =
[[1,1],[1,1],[1,1],[1,1],[1,1],[1,1],[1,1],[1,1,1],[1,1,1,1,1],[1,1],[1,1,1,1
]]
columns =
['Ligand_ID','Compared_Pocket_1','Compared_Pocket_2','Pkt_Ligs_Aligned_RMSD']

results_file = 'Ligs_Align_Pkt_Docked.tsv'
with open(results_file, "w") as outfile:
    writer = csv.writer(outfile, delimiter = "\t")
    writer.writerow(columns)
# loop over each folder
for indx in range(0,len(pkt_counts)):
```

```
    # create columns for the csv file in a dictionary
    all_columns = {}
    for i in range(len(columns)):
        all_columns.setdefault(columns[i],[])
    #
    pkt_comb = combinations(enumerate(pkt_counts[indx]), 2)
    # take each combination of the number of pockets
    for j in list(pkt_comb):

        lig_name1 = "_".join([ligs[indx], (protein[indx])[((j)[0])[0]],
"out"])
        lig_pkt_name1 = ".".join([lig_name1, 'pdbqt'])

        lig_name2 = "_".join([ligs[indx], (protein[indx])[((j)[1])[0]],
"out"])
        lig_pkt_name2 = ".".join([lig_name2, 'pdbqt'])

        # PyMol begins
        cmd.load('%s'%(lig_pkt_name1))
        cmd.load('%s'%(lig_pkt_name2))
        object_list = cmd.get_names()

        rmsd = cmd.align('%s'%(object_list[0]), '%s'%(object_list[1]),
cycles=0, object=None, target_state=0,mobile_state=0)

        # delete loaded ligands before start of the next comparison
        cmd.delete('all')
        # PyMol ends

        compared_pkt_1 = (protein[indx])[((j)[0])[0]]
        compared_pkt_2 = (protein[indx])[((j)[1])[0]]
        all_columns['Ligand_ID'].append(ligs[indx])
        all_columns['Compared_Pocket_1'].append(compared_pkt_1)
        all_columns['Compared_Pocket_2'].append(compared_pkt_2)
        all_columns['Pkt_Ligs_Aligned_RMSD'].append(rmsd[0])

    with open(results_file, "a+") as outfiles:
        writer = csv.writer(outfiles, delimiter = "\t")
        writer.writerows(zip(*[all_columns[key] for key in columns]))
outfiles.close()
```

APPENDIX G: ANALYSIS OF RESULTS

```
# Adding 3D similar Overlap Pocket and AA of Pocket
# Read data and merges them  in a special
# way that assigns the right value to the
# right row and column
#
library(xlsx)
library(readxl)
library(tibble)

# Read data
# data files have the same naming format:
# 1. for docking it is "ligang_name"_AutoDock_vina_Results.tsv
# e.g. AJLF_AutoDock_vina_Results.tsv
```

```
# 2. for pocket it is "ligang_name"_APoc_Binding_pocket_comparison.tsv
# e.g. AJLF_APoc_Binding_pocket_comparison.tsv

ligand <- c("AJLF","CLQV","DTZD","IKSH","NKOP","USZP","XLWJ","YKMS")

data_comb_final <- data.frame()

for (lig in ligand) {
  # Read docking results data
  path_dock <- c("/Users/kwabena/Research/GPCR/AutoDock_ligands_GPCR")
  path_dock <- paste(path_dock, lig, sep = '/')
  file_dock <- paste(lig, c("AutoDock_vina_Results.tsv"), sep = '_')
  setwd(path_dock)
  data_dock <- read.table(file = file_dock, header = T, sep = "\t",
                          stringsAsFactors = F)

  # Read pocket comparison results data
  path_pock <- c("/Users/kwabena/Research/GPCR/Binding_pocket_comparison")
  path_pock <- paste(path_pock, lig, c("APoc"), sep = '/')
  file_pock <- paste(lig, c("APoc_Binding_pocket_comparison.tsv"), sep = '_')
  setwd(path_pock)
  data_pock <- read.table(file = file_pock, header = T, sep = "\t",
                          stringsAsFactors = F)

  # add columns to data_pock for each ligand
  data_pock <- add_column(data_pock, ligand_ID = lig, .before = "pocket_1" )
  data_pock <- add_column(data_pock, pocket_1_Affinity_kcal_mol = NA,
                          .after = "pocket_1" )
  data_pock <- add_column(data_pock, pocket_2_Affinity_kcal_mol = NA,
                          .after = "pocket_2" )

  # extract the affinities for each ligand and add it to data_pock
  pocket_1_pdbID <- substr(data_pock[,2], 10, 13)
  pocket_1_num <- as.numeric(substr(data_pock[,2], 22, 22))
  pocket_2_pdbID <- substr(data_pock[,4], 10, 13)
  pocket_2_num <- as.numeric(substr(data_pock[,4], 22, 22))
  affinity_1 <- c()
  affinity_2 <- c()
  for (row in 1:NROW(data_pock)) {
    cond1 <- (data_dock$protein_PDB_ID == pocket_1_pdbID[row] &
                data_dock$pocket_number == pocket_1_num[row])
    affinity_1[row] <- data_dock[cond1,4]

    cond2 <- (data_dock$protein_PDB_ID == pocket_2_pdbID[row] &
                data_dock$pocket_number == pocket_2_num[row])
    affinity_2[row] <- data_dock[cond2,4]
  }
  # find the adsolute difference of the affinities of the two pockets
  data_pock$abs_diff_aff_1_aff2 <- abs(affinity_1-affinity_2)
  data_pock$pocket_1_Affinity_kcal_mol <- affinity_1
  data_pock$pocket_2_Affinity_kcal_mol <- affinity_2

  # combine data for all ligands
  data_comb_final <- rbind(data_comb_final, data_pock)
}

setwd("/Users/kwabena/Research/GPCR/AutoDock_ligands_GPCR")
```

```
write.table(data_comb_final, file =
"Combine_docking_pocket_comp_results_data.tsv",
            sep = "\t", row.names = F)

setwd("/Users/kwabena/Research/GPCR/AutoDock_ligands_GPCR")
data_comb_final <-
read.table("Combine_docking_pocket_comp_results_data.tsv",sep = "\t",
                              stringsAsFactors = F)

setwd("/Users/kwabena/Research/GPCR/3D_Similar_Coincide_Pkt")
Similar_Coincide_Pkt <- read.table(file = "3D_Similar_Overlap_Pkt.csv",
header = T,
                                   sep = ",", stringsAsFactors = F)
AA_Pkt <- read.table(file = "AA_Pkt.csv", header = T,
                     sep = "\t", stringsAsFactors = F)

setwd("/Users/kwabena/Research/GPCR/AutoDock_ligands_GPCR")
Ligs_Align_Pkt <- read.table(file = "Ligs_Align_Pkt.csv", header = T,
                              sep = "\t", stringsAsFactors = F)

######## 3D similar Overlap Pocket
for (lig in ligand) {
  cond_data_comb <- which(data_comb_final$ligand_ID == lig)
  cond_similar_coincide <- which(Similar_Coincide_Pkt$ligand_ID == lig)
  for (i in cond_similar_coincide) {
    for (j in cond_data_comb) {
      if (substr(data_comb_final[j,2], 10, 13)==
substr(Similar_Coincide_Pkt[i,2], 1, 4)){
        if (substr(data_comb_final[j,4], 10, 13)==
substr(Similar_Coincide_Pkt[i,2], 11, 14)){
          if (substr(data_comb_final[j,2], 10, 13)==
substr(Similar_Coincide_Pkt[i,3], 1, 4)){
            if (substr(data_comb_final[j,2], 22, 22)==
substr(Similar_Coincide_Pkt[i,3], 10, 10)){
              data_comb_final[j,15] <- Similar_Coincide_Pkt[i,4]
              data_comb_final[j,16] <- Similar_Coincide_Pkt[i,5]
            }
          }
        }
      }

      if (substr(data_comb_final[j,2], 10, 13)==
substr(Similar_Coincide_Pkt[i,2], 11, 14)){
        if (substr(data_comb_final[j,4], 10, 13)==
substr(Similar_Coincide_Pkt[i,2], 1, 4)){
          if (substr(data_comb_final[j,4], 10, 13)==
substr(Similar_Coincide_Pkt[i,3], 1, 4)){
            if (substr(data_comb_final[j,4], 22, 22)==
substr(Similar_Coincide_Pkt[i,3], 10, 10)){
              data_comb_final[j,17] <- Similar_Coincide_Pkt[i,4]
              data_comb_final[j,18] <- Similar_Coincide_Pkt[i,5]
            }
          }
        }
      }
    }
  }
}
```

```
######## Alignment of ligands for each pockets compared
for (lig in ligand) {
  cond_data_comb <- which(data_comb_final$ligand_ID == lig)
  cond_ligs_align <- which(Ligs_Align_Pkt$ligand_ID == lig)
  for (i in cond_data_comb) {
    for (j in cond_ligs_align) {
      if (substr(data_comb_final[i,2], 10, 13)== substr(Ligs_Align_Pkt[j,2],
1, 4)){
        if (substr(data_comb_final[i,2], 22, 22)==
substr(Ligs_Align_Pkt[j,2], 6, 6)){
          if (substr(data_comb_final[i,4], 10, 13)==
substr(Ligs_Align_Pkt[j,2], 8, 11)){
            if (substr(data_comb_final[i,4], 22, 22)==
substr(Ligs_Align_Pkt[j,2], 13, 13)){
              data_comb_final$Pkt_ligs_align_rmsd[i] <-
Ligs_Align_Pkt$RMSD[j]
              break
            }
          }
        }
      }
    }
  }
}

######## AA of Pocket
for (ligs in ligand) {
  cond_data_comb_2 <- which(data_comb_final$ligand_ID == ligs)
  cond_AA_Pkt <- which(AA_Pkt$ligand_ID == ligs)
  for (s in cond_AA_Pkt) {
    for (n in cond_data_comb_2) {

      if (substr(data_comb_final[n,2], 10, 13) == AA_Pkt[s,2]){
        if (substr(data_comb_final[n,2], 22, 22) == AA_Pkt[s,3]){
          data_comb_final$GPCR_Resi_pkt_1[n] <- AA_Pkt[s,4]
        }
      }
      if (substr(data_comb_final[n,4], 10, 13) == AA_Pkt[s,2]){
        if (substr(data_comb_final[n,4], 22, 22) == AA_Pkt[s,3]){
          data_comb_final$GPCR_Resi_pkt_2[n] <- AA_Pkt[s,4]
        }
      }

    }
  }
}

# Correlation analysis
corr_table <- data.frame(row.names = F)
for (i in 1:length(ligand)) {
  cond <- which(data_comb_final$ligand_ID == ligand[i])
  data <- data_comb_final[cond,]
  corr <- cor.test(data$PS_score, data$abs_diff_aff_1_aff2,
                   alternative = c("two.sided"), method = c("pearson"),
                   conf.level = 0.95)
  corr_list <- list(Ligand = ligand[i],
```

```
Correlation_PS_score_abs_diff_aff1_aff2_pockets=as.vector(corr$estimate),
                    P_value =corr$p.value)
  corr_table <- rbind(corr_table, corr_list, stringsAsFactors = F)
}

setwd("/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control")
data_control <- read_excel("Combined_Data.xlsx")

colnames(data_control)

######## Convert AA 3 letter code to 1 letter code
AA <-
c("ALA"="A","ARG"="R","ASN"="N","ASP"="D","CYS"="C","GLU"="E","GLN"="Q","GLY"
="G","HIS"="H",

"ILE"="I","LEU"="L","LYS"="K","MET"="M","PHE"="F","PRO"="P","SER"="S","THR"="
T","TRP"="W",
        "TYR"="Y","VAL"="V")
data_control$Protein_AA_Interact_pkt_1 <-
as.character(data_control$Protein_AA_Interact_pkt_1)
data_control$Protein_AA_Interact_pkt_2 <-
as.character(data_control$Protein_AA_Interact_pkt_2)
data_control$Resi_pkt_1 <- as.character(data_control$Resi_pkt_1)
data_control$Resi_pkt_2 <- as.character(data_control$Resi_pkt_2)

for (k in 1:NROW(data_control)) {
  ####### Begin: Residues that are involved in interaction
  if (data_control$Protein_AA_Interact_pkt_1[k] != "-"){
    pkt_1 <- unlist(strsplit(data_control$Protein_AA_Interact_pkt_1[k], split
= ","))
    pkt_1_seq = ""
    for (l in 1:length(pkt_1)) {
      pkt_1_seq <- paste0(pkt_1_seq, AA[toupper(pkt_1[l])])
    }
    if (nchar(pkt_1_seq) == length(pkt_1)){
      data_control$AA_Interact_pkt_1[k] <- pkt_1_seq
    }
  }else{
    data_control$AA_Interact_pkt_1[k] <- "-"
  }

  if (data_control$Protein_AA_Interact_pkt_2[k] != "-"){
    pkt_2 <- unlist(strsplit(data_control$Protein_AA_Interact_pkt_2[k], split
= ","))
    pkt_2_seq = ""
    for (m in 1:length(pkt_2)) {
      pkt_2_seq <- paste0(pkt_2_seq, AA[toupper(pkt_2[m])])
    }
    if (nchar(pkt_2_seq) == length(pkt_2)){
      data_control$AA_Interact_pkt_2[k] <- pkt_2_seq
    }
  }else{
    data_control$AA_Interact_pkt_2[k] <- "-"
  }
  ####### End: Residues that are involved in interaction
```

```
  ####### Begin: Residues in the pocket
  pkts_1 <- unlist(strsplit(data_control$Resi_pkt_1[k], split = ","))
  pkts_1_seq = ""
  for (t in 1:length(pkts_1)) {
    pkts_1_seq <- paste0(pkts_1_seq, AA[toupper(pkts_1[t])])
  }
  if (nchar(pkts_1_seq) == length(pkts_1)){
    data_control$AA_pkt_1[k] <- pkts_1_seq

  }

  pkts_2 <- unlist(strsplit(data_control$Resi_pkt_2[k], split = ","))
  pkts_2_seq = ""
  for (u in 1:length(pkts_2)) {
    pkts_2_seq <- paste0(pkts_2_seq, AA[toupper(pkts_2[u])])
  }
  if (nchar(pkts_2_seq) == length(pkts_2)){
    data_control$AA_pkt_2[k] <- pkts_2_seq
  }
  ####### End: Residues in the pocket
}

######## Create more features
library(Peptides)
for (v in 1:NROW(data_control)) {
  ################################################ Pkt 1
  seqs1 <- data_control$AA_pkt_1[v]

  # MS-WHIM scores of a protein sequence: MS-WHIM scores were derived from 36
  # electrostatic potential properties derived from the three- dimensional
  # structure of the 20 natural amino acids
  c = mswhimScores(seqs1)[[1]][c(1,2,3)]
  data_control$Pkt_1_MSWHIM1[v] <- c[1]; data_control$Pkt_1_MSWHIM2[v] <-
c[2];
  data_control$Pkt_1_MSWHIM3[v] <- c[3];

  ################################################ Pkt 2
  seqs2 <- data_control$AA_pkt_2[v]

  # MS-WHIM scores of a protein sequence: MS-WHIM scores were derived from 36
  # electrostatic potential properties derived from the three- dimensional
  # structure of the 20 natural amino acids
  cc = mswhimScores(seqs2)[[1]][c(1,2,3)]
  data_control$Pkt_2_MSWHIM1[v] <- cc[1]; data_control$Pkt_2_MSWHIM2[v] <-
cc[2];
  data_control$Pkt_2_MSWHIM3[v] <- cc[3];

}

# MSWHIM distance

lf_dist_MSWHIM <- function(x,y){
  sum_abs_diff_mswhim <- max(abs(x[1]-y[1]), abs(x[2]-y[2]), abs(x[3]-y[3]))
  sum_abs_diff_mswhim
}

colnames(data_control)
for (h in 1:NROW(data_control)) {
```

```
  x <- unlist(as.vector(data_control[h,c(21,22,23)]))
  y <- unlist(as.vector(data_control[h,c(24,25,26)]))
  data_control$Lf_Dist_MSWHIM[h] <- lf_dist_MSWHIM(x,y)
}

# Affinity
l1_dist_Affinity <- function(x,y){
  sum_abs_diff_Affinity <- abs(x-y)
  sum_abs_diff_Affinity
}

colnames(data_control)
for (h in 1:NROW(data_control)) {
  x <- unlist(as.vector(data_control[h,c(15)]))
  y <- unlist(as.vector(data_control[h,c(16)]))
  data_control$L1_Dist_Affinity[h] <- l1_dist_Affinity(x,y)
}


# Num_Same_Resi_Interact
for (s in 1:NROW(data_control)) {
  string <- strsplit(c(data_control$Protein_AA_Interact_pkt_1[s],
                       data_control$Protein_AA_Interact_pkt_2[s]), ",")
  count_aa <- 0
  if(length(string[[1]]) <= length(string[[2]])){
    for (t in string[[1]]) {
      indx <- which(string[[2]] == t)
      if (length(indx) >= 1){
        count_aa = count_aa + 1
        string[[2]] <- string[[2]][-c(indx[1])]
      }
    }
  }else{
    for (t in string[[2]]) {
      indx <- which(string[[1]] == t)
      if (length(indx) >= 1){
        count_aa = count_aa + 1
        string[[1]] <- string[[1]][-c(indx[1])]
      }
    }
  }
  data_control$Num_Same_Resi_Interact[s] <- count_aa
}

# P3D_similar_Coincide_pkt
data_control$Average_P3D_similar_Coincide_pkt_1 <-
0.5*(data_control$P3D_similar_Coincide_pkt_1_flex +

data_control$P3D_similar_Coincide_pkt_1_rigid)
data_control$Average_P3D_similar_Coincide_pkt_2 <-
0.5*(data_control$P3D_similar_Coincide_pkt_2_flex +

data_control$P3D_similar_Coincide_pkt_2_rigid)

data_control$Sum_P3D_similar_Coincide_pkts_pair <-
(data_control$Average_P3D_similar_Coincide_pkt_1 +

data_control$Average_P3D_similar_Coincide_pkt_2)
```

```
######## Write data to excel
library(xlsx)
setwd("/Users/kwabena/Research/GPCR/Manuscript/MDPI
biomolecules/New_Data_Control")
write.table(data_control[,c(1:5,8,15,16,28)], file =
"Control_Combined_Data.tsv",sep = "\t",
            col.names = TRUE, row.names = FALSE, append = FALSE)

######### Plot

# Actual_pkt_ligs_aligned_RMSD vs Docked_pkt_ligs_aligned_RMSD
plot(data_control$Actual_Pkt_Ligs_Aligned_RMSD, type="l",col="red",
     ylab = "RMSD of Aligned Ligands in Pairs of Pockets", xlab = "Index of
Pairs of Pockets")
lines(data_control$Docked_Pkt_Ligs_Aligned_RMSD, col="blue", type = "l")
legend(1, 3, legend=c("RMSD Acutal", "RMSD Docked"),
       col=c("red", "blue"), lty=c(1,1), cex=0.8)

######### Correlation Analysi
# PS_score vs:
# Actual_pkt_ligs_aligned_RMSD
# Lf_Dist_MSWHIM
# Num_Same_Resi_Interact
# L1_Dist_Affinity
# Sum_P3D_similar_Coincide_pkts_pair

# PS_score
mean(data_control$PS_score)
sd(data_control$PS_score)
min(data_control$PS_score)
max(data_control$PS_score)


# Actual_pkt_ligs_aligned_RMSD --- Negative and Significant
corr <- cor.test(data_control$PS_score,
data_control$Actual_Pkt_Ligs_Aligned_RMSD,
                 conf.level = 0.95, alternative = c("two.sided"), method =
c("pearson"))
corr

# Actual_pkt_ligs_aligned_RMSD vs Docked_pkt_ligs_aligned_RMSD ---
Significant
var.test(x=data_control$Actual_Pkt_Ligs_Aligned_RMSD,
y=data_control$Docked_Pkt_Ligs_Aligned_RMSD,
         alternative = "two.sided")
t.test(x=data_control$Actual_Pkt_Ligs_Aligned_RMSD,
y=data_control$Docked_Pkt_Ligs_Aligned_RMSD,
       alternative = "two.sided", var.equal = F, paired = T)

mean(data_control$Actual_Pkt_Ligs_Aligned_RMSD)
sd(data_control$Actual_Pkt_Ligs_Aligned_RMSD)

mean(data_control$Docked_Pkt_Ligs_Aligned_RMSD)
sd(data_control$Docked_Pkt_Ligs_Aligned_RMSD)

# L1_Dist_Affinity --- Negative and Significant
```

```
corr <- cor.test(data_control$PS_score, data_control$L1_Dist_Affinity,
                 alternative = c("two.sided"), method = c("pearson"),
conf.level = 0.95)
corr

# Ignore
# Sum_P3D_similar_Coincide_pkts_pair --- Positive but Not Significant
corr <- cor.test(data_control$PS_score,
data_control$Sum_P3D_similar_Coincide_pkts_pair,
                 alternative = c("two.sided"), method = c("pearson"),
conf.level = 0.95)
corr

# Ignore
# Lf_Dist_MSWHIM --- Negative and Significant
corr <- cor.test(data_control$PS_score, data_control$Lf_Dist_MSWHIM,
                 alternative = c("two.sided"), method = c("pearson"),
conf.level = 0.95)
corr

# Ignore
# Num_Same_Resi_Interact --- Positive but Not Significant
corr <- cor.test(data_control$PS_score, data_control$Num_Same_Resi_Interact,
                 alternative = c("two.sided"), method = c("pearson"),
conf.level = 0.95)
corr
```

Curriculum Vita

Kwabena Owusu Dankwah was born in Ghana to Mr. Yaw Karikari Dankwa and Madam Mary Darkwa. He is the youngest child of his parents. He got graduated from Oda Senior High School at Oda, Ghana, in 2009. He entered University of Cape Coast (UCC) in fall 2010 to pursue his bachelor's degree in Statistics. After graduating in 2014, he served as a Teaching Assistant at the same institution (UCC). In the fall of 2016, he entered the Graduate School of New Mexico State University (NMSU) at Las Cruces, New Mexico. During his studies at NMSU, he worked as a Teaching Assistant, and as the Laboratory Instructor for the institution. Kwabena got accepted in the Computational Science program at University of Texas at El Paso to pursue his Doctoral degree in fall 2018 and currently resides in El Paso.

Email address: okdankwah@miners.utep.edu & owusukd@yahoo.com