**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.**

**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”*

* 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”
* 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”
* Data on GPCR regional (N-terminal, extracellular loops, intracellular loops, the seven helices, and the C-terminal) positions – Need to automate this

*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**
* 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.
* 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”*

* 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”*

* 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

* 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

* 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 ”*

* 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”*
* 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”*

* 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”*

* 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**

* 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”*

* For GPCRs that bind the same ligand: - needs to automate this
* 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**

* 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*”