Chapter 3: Computational Methodology

Table of Contents

3.1 Abstract 3

3.2 Introduction 4

3.3 Studying membrane protein folding and structure 5

3.3.1 Rosetta 5

3.3.2 Molecular Software Library 5

3.3.3 Docking Algorithms 6

3.3.4 Machine learning tools 6

3.4 Protein Design Algorithm 8

3.4.1 Analysis of membrane protein PDBs 8

3.4.2 Defining the Interface 9

3.4.3 Developing the energy terms 9

3.4.4 Sequence Search 11

3.4.5 Backbone Refinement 12

3.4.6 Mutating the interface 12

3.5 Analysis 13

3.5.1 Software 13

3.5.2 Design Analysis 13

3.5.3 Next Generation Sequencing Fluorescence Reconstruction 14

3.5.4 TOXGREEN Conversion 15

3.5.5 Trimming for Clashing Mutations 16

3.5.6 Hydrogen Bond Mutations 16

3.6 Conclusion 16

3.7 References 16

3.1 Abstract

With recent advancements in experimental techniques, researchers are more easily able to design experiments to retrieve high-throughput data. In conjunction with high-throughput data, computational tools and software have been developed to complement high-throughput data collection. However, understanding how to use and interpret these computational tools is a necessary skill to properly interpret and analyze high-throughput data. An important approach to designing and distributing these computational tools is the ability to explain how these technologies work to efficiently share them technologies with the research community (Greener et al., 2022; Mougeot et al., 2022; Na, 2020; van Iterson et al., 2012). The ability to assess proteins structures allows researchers to determine how and why proteins fold with unique structural features essential for function. Many computational algorithms have been developed as tools to predict and design unknown proteins using information from solved structures. In this section, I will review a variety of computational tools that have been developed to expand understanding of protein structures, in particular highlighting commonly used structure prediction tools such as Rosetta and Alphafold. I then discuss in detail how I developed my protein design algorithm and the other programs used in my research.

3.2 Introduction

The protein data bank (PDB) was established to collaborate and share protein structures globally online. This tool allows researchers to deposit solved protein structures for others to access and assess their findings (Berman et al., 2000). Initially, proteins structures were solved primarily using x-ray crystallography. X-ray crystallography has contributed to solving ~80% of MP structures (Kermani, 2021). However, producing crystals at high resolution remains a difficulty in crystallography. To bypass the need to crystallize proteins, many other MP structures have been solved by nuclear magnetic resonance (NMR). Solid-state NMR has bypassed the need for detergents in crystallography, obtaining structures of MPs less than 50 residues within lipid bilayers or nanodiscs (Liang & Tamm, 2016).

The complex nature of the lipid bilayer makes it difficult to directly study the forces involved in MP folding. As an initial approach, researchers aim to determine the structures of MPs to identify structural features important for folding. However, solving MP structures is an inherently difficult task due to the need to express and solubilize MPs for experiments. Furthermore, many of these approaches cannot determine more than a single protein structure, making it difficult to use this information to understand dynamic structural changes involved in folding. More recently cryo-EM has been used to solve MP structures. In addition to bilayers and nanodiscs, it is possible to solubilize and obtain the structures of MPs within detergents, saposin-lipoprotein nanoparticles, amphipols, and peptidiscs (Januliene & Moeller, 2021). Cryo-EM allows MP structures to be studied in a large variety of different environments, giving researchers the ability to study alternative structures of these proteins by changing the solubilization conditions. Despite the advancements in MP structural characterization, many of these efforts take years to determine conditions that yield publishable results in high resolution. MPs make up only 4.6% of structures deposited in the PDB (April 2024; PDB). Lacking MP structures to assess, some groups of researchers focus on using known protein structures to develop algorithms that can be used to predict or design unknown proteins. These algorithms are then often used as tools to estimate the contribution that a variety of biophysical forces have on MP stability.

3.3 Studying membrane protein folding and structure

In conjunction with structural determination, computational methods have been invented by evaluating previously solved MP structures. Many of these methods look to further understand MP folding by establishing energetic terms that aim to estimate the thermodynamics of MP folding. Below I review the current computational methods used to predict protein structures, highlighting unique features and energetics of each tool.

3.3.1 Rosetta

David Baker’s Lab at University of Washington are experts in this area, utilizing known energetic and structural information to develop the software suite Rosetta. Rosetta houses a variety of energy functions and prediction tools for soluble environments, including the ability to dock or design proteins *de novo*. **(add in the energies here)** The original Rosetta energy functions have been adapted to accommodate predicting helical TMs within the membrane environment. The updated functions include an energy term that models the membrane into layers classified defining atoms as water-exposed, polar, interface, and hydrophobic (White & Wimley, 1999; Yarov-Yarovoy, Schonbrun, et al., 2006). Successful predictions determined structural details in MPs associated with voltage sensing and gating mechanisms (Vargas et al., 2012; Yarov-Yarovoy, Baker, et al., 2006). Recently, RosettaMP was developed to enhance the functionality of MP prediction which includes modeling TM helices *de novo* from sequence, a representation of the membrane bilayer, and the ability to transform a protein into the membrane (Koehler Leman et al., 2017). These tools increase the accessibility to MP structure, improving the ability to visualize and predict structures of MPs that have not yet been solved. Simultaneously, energetic predictions enhance the knowledge in the field by allowing researchers to assess structural mutations *in silico* prior to testing with *in vitro* or *in vivo* experiments.

3.3.2 Molecular Software Library

Another computational tool for membrane proteins is the Molecular Software Library, or MSL (Kulp et al., 2012). Similar to Rosetta, MSL houses a variety of tools to perform MP structure prediction. These tools include the ability to transform proteins in space, mutate specific residues, extract geometric information from known structures, and predict the structure and energetics of an input sequence. Additionally, it has been used to develop the CATM prediction algorithm. The CATM algorithm uses the Energy-Based conformer library applied at the 95% level for side chain mobility (Subramaniam & Senes, 2012). Energetics of predicted proteins are determined using the CHARMM 22 van der Waals function (MacKerell et al., 1998), the IMM1 membrane implicit solvation model (Lazaridis, 2003), and the hydrogen bonding function of SCWRL4 (Krivov et al., 2009). Each of these energy terms is used to optimize the dimer geometry by Monte Carlo backbone perturbation cycles where all parameters (xShift, zShift, axialRotation, and crossingAngle) are locally varied. The predicted energy of association is calculated as the energy of the dimer minus the energy of two monomers.

The CATM algorithm has been applied to predict the structures of both known and unknown TM dimers, and experimental studies have shown that it accurately predicts the energetics of association (Anderson et al., 2017; Díaz Vázquez et al., 2023; Mueller et al., 2014). My research adapts the CATM algorithm to design structures with strong van der Waals packing in the absence of hydrogen bonding, allowing me to assess the extent at which packing can drive MP association. I further detail how I used MSL to design my TM homodimers in section 3.4.

3.3.3 Docking Algorithms

Other methods for predicting interactions between MPs focus on predicting the topology or docking of individual TM helices. OCTOPUS predicts TM topology using a combination of Markov models and neural networks (Viklund & Elofsson, 2008). HADDOCK can apply experimental knowledge of the interface region between proteins to refine docking (de Vries et al., 2010; Dominguez et al., 2003). PREDDIMER utilizes a novel surface-based modeling approach to predict and screen TM dimers for conformation heterogeneity (Polyansky et al., 2012). EVFold uses evolutionary based structural restraints to refine their docked structures (Braun T et al., 2015). TMDOCK uses an all-atom model for helices, inserting them in the membrane and outputting a structure alongside a predicted ΔG of insertion and ΔG of association (Lomize & Pogozheva, 2017). Each of these methods was adapted into a webserver for online use, where users can input the sequence and additional information to guide the process.

3.3.4 Machine learning tools

Most notably, at the 2020 Critical Assessment of Structure Prediction (CASP) conference, Google’s Deepmind introduced the machine learning model AlphaFold. Unlike previously mentioned prediction algorithms, AlphaFold predicts structures without energetics. AlphaFold utilizes a combination of neural networks and training on multiple sequence alignments (MSAs) and solved protein structures to predict unknown structures to near atomic precision with a 95% confidence interval (Jumper et al., 2021). Shortly afterward, David Baker’s group introduced RoseTTAFold, improving on the Rosetta prediction by incorporating a similar architecture to AlphaFold, with the addition of a third track network that connects sequence, residue-residue distances, and atomic coordinates (Baek et al., 2021). Each of these methods drastically improved the ability to predict unknown protein structures using information from previously studied and solved proteins. With increasing interest in using these technologies, multiple free webservers have been established to enhance access to these advanced protein prediction algorithms (Mirdita et al., 2022; Roberts et al., 2024). However, there are limits to how well these prediction algorithms work. AlphaFold struggles to predict proteins with <30 homologs in their MSAs, and accuracy decreases for protein structures dependent on other protein-protein interactions, while RoseTTAFold has difficulty predicting higher-order oligomers (Agard et al., 2022). The disparity between soluble proteins and MPs in the PDB contributes to these limitations, making small TM proteins difficult to predict. To better understand the dynamics of association and folding in MPs, it is necessary to advance our knowledge of the forces involved in folding.

3.4 Protein Design Algorithm

To investigate the impact of van der Waals packing on membrane protein association, I opted for a high-throughput design approach. I developed a sequence search algorithm that allowed me to design 1000s of membrane protein structures using MSL v. 1.1, an open source C++ library that is freely available at http://msl-libraries.org (Kulp et al., 2012). I coupled this algorithm with structural backbone refinement also built in MSL. Below, I detail the algorithm alongside the experiments and tests to develop it.

3.4.1 Analysis of membrane protein PDBs

A diagram of a protein distribution

Description automatically generated with medium confidence

Using a program developed in MSL, I extracted backbone geometries from all unique MPs found in the Orientations of Membrane Proteins database (OPM) (Lomize et al., 2006). This program reads an input PDB structure, identifies helical subunits of the protein composed of at least 13 AAs, and measures the distances between Cα carbons. Any two helices with at least 3 Cα carbons within 9Å of each other are extracted as individual helical pairs. To ensure that I did not extract helical pairs from redundant structures, the MPs from OPM were trimmed by sequence similarity. The PDB clusters protein sequences weekly using MMseqs2 with a variety of different sequence similarity cutoffs (Steinegger & Söding, 2017). I trimmed the OPM dataset for unique structures based on 30% sequence similarity from the PDB and compiled the geometric data. I then extracted two global parameters the crossingAngle and xShift, which were plotted against each other for each protein and analyzed using kernel density estimation.

**(add in how I got the other parameters here; explain and define each of the regions)**

3.4.2 Defining the Interface

A video game screen with a black background

Description automatically generatedTo reduce heterogeneity in expression, I designed the interface of a standardized TM helix of 21 AAs consisting of a poly-Leu backbone, a strategy previously used to study the association of GASrights (Anderson et al., 2017). In my first protein design run, I used Solved Accessible Surface Area (SASA) to identify the interfacial positions of a protein set at a geometry from my membrane protein analysis. I calculated the SASA for each position on the dimer and defined the interface as the 8 positions with the least amount of access to the solvent, or the most buried positions. However, after sorting these designs and looking at the fluorescence results, variations in the interface were seen to increase the complexity of the data (S1). **(talk about the similarities in association by proteins with similar interfaces)**

After seeing these results, I standardized the interface for my future design runs. The GASright was previously studied using similarly standardized backbones, so we chose to use the same 8 positions on the interface. Since right handed designs have similar geometric parameters outside of a larger xShift, I decided to use the same 8 positions as GASright. For the left handed designs, I visually inspected a variety of sequences with unique crossing angles and defined 3 interfaces. These interfaces allowed for me to account for known left-handed association motifs such as the leucine zipper and coiled-coil (cite). Standardizing the interfaces allowed me to come up with a consistent mutational strategy to assess my proteins for their association at the given interface that is discussed in detail in section 3.4.6.

3.4.3 Developing the energy terms

To assess my designed proteins, I used the same set of energy terms used previously in CATM. These terms are calculated for each protein during the interfacial sequence design search to find the best interfacial sequence for the input geometric structure. To determine the stability of our designed dimers, we predict the interaction energies of the dimer structure and two monomers, and then subtract the monomer energy from the dimer energy (Dimer-Monomer). However, computing the monomer energy for each sequence initially created a bottleneck in our design algorithm, limiting the number of sequences we could design. To account for this, I developed an energy term that estimates the monomer energy of each sequence: BASELINE\_MONOMER.

A graph with a line graph

Description automatically generated with medium confidence

The BASELINE\_MONOMER term was created by measuring the energetic contributions from each individual amino acid. These energy contributions were measured using the following energy terms: CHARMM\_VDW, SCWRL4\_HBOND, and CHARMM\_IMM1. I computed the energetics of 10000 random monomer sequences and measured the self and pair energies for each amino acid. The self-energy is the energy contribution for an individual amino acid to the protein stability alone, while the pair energies are the energy contribution between any two interacting amino acids (cite). I measured the pair energies for all pairs on the sequence and found that there was no longer a pair interaction found between amino acids more than 10 away from each other. From each iterations, I calculated the average of all self and pair energies and saw a strong correlation between the measured monomer energy and the BASELINE\_MONOMER term (figure). This BASELINE\_MONOMER term was developed for the subset of amino acids (G, A, L, I, M, S, T, W, Y, F) and would need to be rerun to establish a term for any additional amino acids. This term was helpful in decreasing computational time, allowing us to design 1000s of sequences within a week.

Another issue that I encountered while designing my proteins was that many of my initial designs were often composed of only 2-3 different AAs. This result could impact our protein expression, where designs with diverse sequences do not express similarly to designs composed of a few AAs. To combat this, I developed a SEQUENCE\_ENTROPY term that outputs a score based on how similar the output sequence is to a typical membrane protein sequence. This term was developed by using the sequences extracted from membrane protein PDBs and defining the frequency of each AA.

To convert the frequency of AAs in a sequence to an energy term, I utilized the following equation:

First, the number of each AA (#AA) is counted within the sequence. Then, the number of possible permutations for the sequence to occur is determined using the following equation:

where n is the number of positions, which is divided by the factorial for the number of each AA in the sequence multiplied. Finally, the probability of the sequence occurring in membrane proteins is calculated using the frequency of each AA in membrane protein sequences (freq\_AA) to the power of the number of each AA in the sequence multiplied by the permutations:

The probability is then inserted into the initial equation, returning a value that can be applied as an additional energy term for each sequence.

3.4.4 Sequence Search

For each input geometry, the algorithm first defines the best sequence using the Self Consistent Mean Field (SCMF) theorem (cite). The sequence from the SCMF is then run through a Monte Carlo sequence optimization: A random position on the interface is switched to a random AA from the given set of AAs, and the energy is calculated using the combination of defined energies in 3.3.3. Each energy term can be multiplied by an optional weight, allowing the total energy to be changed proportionately to the given weights. This allows for versatility in the protein design process. The equation for the total energy is as follows:

After optimization and testing, we found that the SEQUENCE\_ENTROPY term results in diverse membrane protein sequences at a weight of 10. This weight allows the SEQUENCE\_ENTROPY term to play a significant role in the interface sequence search. The total energy is used for the acceptance criteria, where early in the MC more leeway is given, allowing the algorithm to search for a variety of sequences in energetic space before reaching an energetic minimum. The sequences are saved in an output file to show the trajectory of sequences that were chosen along with their corresponding energy terms. The interface sequence with the best interaction energy is saved as the design undergoes further backbone refinement. **(a figure here would be nice)**

3.4.5 Backbone Refinement

Another improvement from my initial design run was to use backbone iterative refinement, inspired by Rosetta. After the best sequence is found, I implemented a MC based structural refinement procedure to find the best structure for the sequence. During the refinement process, a random geometric term is chosen and shifted as done previously in CATM. The total interaction energy for the refined structure is again used as the acceptance criteria, with the BASELINE\_MONOMER energy being replaced by the calculated monomer energy and the SEQUENCE\_ENTROPY term no longer used:

This new geometry can then be used as an input geometry for the sequence search and backbone refinement process to determine if there are other designable sequences with this similar geometry.

3.4.6 Mutating the interface

For each protein, I made point mutations on the interface corresponding to interfacial positions expected to decrease association. I created a program that accepts the structure and interfacial positions as inputs, individually mutates the positions to a given amino acid, and outputs the energy. The first mutation is a clash mutant, where the interface was mutated to an Ile to disrupt the ability for the protein to associate. The second mutation is a void mutant, where the interface was mutated to an Ala to assess the loss of steric packing on association. When comparing the clash mutations to the fluorescence of the designed structure, there is a significant decrease in their association, suggesting that many of my proteins associate by the designed interface. However, the void mutations did not show this same decrease, suggesting that mutating for loss of packing may result in the formation of an alternate interface for association.

I also wanted to have a way to measure the energies of the chosen mutants for each of my structures. To do so, I created a more intensive structural refinement program based on the backbone refinement from the original design program and CATM. The program takes in the given structure and repeats the MC based structural refinement, but this time procedurally decreases by multiplying the previously used upper limit for a geometry by the MC temperature change ratio. For example: As the geometric distance between helices is made shorter, the max distance is decreased, allowing for finer movements to be made. This intensive refinement can be repeated as many times as desired, and the energy output is what I used to evaluate my structures against their reconstructed fluorescence in sort-seq. **(another figure could be good here)**

3.5 Analysis

3.5.1 Software

The following calculations, analyses, and graphing was implemented and performed using Python v. 2.7. Relevant packages include:

Pandas: (McKinney, 2011)

DNAChisel: (Zulkower & Rosser, 2020)

Numpy: (McKinney, 2012)

Matplotlib: (Tosi, 2009)

Seaborn: (Waskom, 2021)

Scipy: (Virtanen et al., 2020)

Sklearn: (Pedregosa et al., 2011)

Pymol: (DeLano, 2002)

All programs and code can be found on Github.

3.5.2 Design Analysis

To analyze my design data, I created a program that compiles the data from all the design runs and outputs a variety of helpful plots for interpreting the design runs. These plots include energy plots against the geometry and AA percent composition against the original frequency found in membrane proteins **(supplemental figures here)**. These sequences alongside their structures can then be input into the mutation program to identify mutations expected to decrease association. Finally, these designed sequences, their respective mutants, and a variety of control sequences were ordered in an oligo pool library from Twist Bioscience and cloned into plasmids for TOXGREEN sort-seq.

3.5.3 Next Generation Sequencing Fluorescence Reconstruction

As detailed previously in (cite sort-seq), plasmids obtained from sorted populations of *E. coli* can be sent for Next Generation Sequencing (NGS) and the sequencing results reconstructed to fluorescence corresponding to dimerization propensity. The NGS results were analyzed using a custom Perl program and then analyzed using Python programs. Reconstructed GFP levels were calculated as a weighted average (Kosuri et al., 2013). This method normalizes the reads per protein per bin with the fraction of the population found in that bin. The normalized fractional contribution of each bin (j) for each protein (i), aij is calculated as:

A diagram of a function

Description automatically generatedA mathematical equation with black letters

Description automatically generated with medium confidence

where the normalized fractional contribution is then multiplied by the median fluorescence of that bin (mj­). Finally, the contributions for each sorted bin are summed to determine the reconstructed fluorescence.

3.5.4 TOXGREEN Conversion

A group of graphs with lines

Description automatically generated with medium confidenceAfter running TOXGREEN sort-seq, the sequencing data is run through a pipeline that reconstructs the fluorescence profile for each sequence found in the dataset (cite sort-seq). However, the reconstructed fluorescence is quite similar between many sequences, and we were unable to determine differences between our sequences. To account for this, we plot the reconstructed fluorescence versus the TOXGREEN fluorescence of the control sequences and a subset of design sequences that we ran in TOXGREEN (figure). The slope and y-intercept of the correlation is used to convert the reconstructed fluorescence of all other sequences to the fluorescence seen in low throughput TOXGREEN. This allows us to interpret differences between sequences using traditional TOXGREEN values.

3.5.5 Trimming the data for proper insertion

A liquid maltose growth assay was run in triplicate as in (cite sort-seq). These populations are sent along with the sorted populations for NGS, and the counts of each sequence can be used to evaluate their ability to properly insert into the membrane. Cloned alongside our designed sequences are control sequences that are known not to insert in the membrane, as shown by failure to grow on maltose plates. Sequences that are more abundant than these controls are considered properly inserted. The relative abundance from the overnight growth (0H) and the growth in liquid maltose at 30H to determine the ability to insert:

Could create a figure similar to SMA with the red bars and the blue to represent good growth and controls and such?

3.5.6 Trimming data for Clashing Mutations

To determine if sequences are associating according to the designed interface, we sought to identify sequences where the clash mutant results in a significant decrease in association. We used a cutoff for monomeric sequences **(fluorescence)** and also accepted any sequences where the mutation resulted in >50% of the design fluorescence in terms of %GpA. **(explain more).** Resulted in x out of y designs (%) **(also show the GAS, right, left splits; maybe as a figures)**

3.5.7 Hydrogen Bond Mutations

To better assess whether packing is the sole force playing a role in the association of our designed proteins, we decided to mutate any proteins with AAs that could result in hydrogen bonding.

3.6 Conclusion

* Talk about ways to improve the molecular software library and/or to transition some of these scripts to something more usable like Rosetta/Alphafold/how to make this easy to use
  + Creating a webserver was a pipe dream I had but never got the skill to do it

3.7 References

Agard, D. A., Bowman, G. R., DeGrado, W., Dokholyan, N. V., & Zhou, H. X. (2022). Solution of the protein structure prediction problem at last: crucial innovations and next frontiers. *Fac Rev*, *11*, 38. <https://doi.org/10.12703/r-01-0000020>

Anderson, S. M., Mueller, B. K., Lange, E. J., & Senes, A. (2017). Combination of Cα-H Hydrogen Bonds and van der Waals Packing Modulates the Stability of GxxxG-Mediated Dimers in Membranes. *J Am Chem Soc*, *139*(44), 15774-15783. <https://doi.org/10.1021/jacs.7b07505>

Baek, M., DiMaio, F., Anishchenko, I., Dauparas, J., Ovchinnikov, S., Lee, G. R.,…Baker, D. (2021). Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, *373*(6557), 871-876. <https://doi.org/10.1126/science.abj8754>

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H.,…Bourne, P. E. (2000). The protein data bank. *Nucleic acids research*, *28*(1), 235-242.

Braun T, Koehler Leman J, & OF, L. (2015). **Combining Evolutionary Information and an Iterative Sampling Strategy for Accurate Protein Structure Prediction**. In: PLOS Computational Biology.

de Vries, S. J., van Dijk, M., & Bonvin, A. M. (2010). The HADDOCK web server for data-driven biomolecular docking. *Nat Protoc*, *5*(5), 883-897. <https://doi.org/10.1038/nprot.2010.32>

DeLano, W. L. (2002). Pymol: An open-source molecular graphics tool. *CCP4 Newsl. Protein Crystallogr*, *40*(1), 82-92.

Dominguez, C., Boelens, R., & Bonvin, A. M. (2003). HADDOCK: a protein-protein docking approach based on biochemical or biophysical information. *J Am Chem Soc*, *125*(7), 1731-1737. <https://doi.org/10.1021/ja026939x>

Díaz Vázquez, G., Cui, Q., & Senes, A. (2023). Thermodynamic analysis of the GAS. *Biophys J*, *122*(1), 143-155. <https://doi.org/10.1016/j.bpj.2022.11.018>

Greener, J. G., Kandathil, S. M., Moffat, L., & Jones, D. T. (2022). A guide to machine learning for biologists. *Nat Rev Mol Cell Biol*, *23*(1), 40-55. <https://doi.org/10.1038/s41580-021-00407-0>

Januliene, D., & Moeller, A. (2021). Single-Particle Cryo-EM of Membrane Proteins. *Methods Mol Biol*, *2302*, 153-178. <https://doi.org/10.1007/978-1-0716-1394-8_9>

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O.,…Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583-589. <https://doi.org/10.1038/s41586-021-03819-2>

Kermani, A. A. (2021). A guide to membrane protein X-ray crystallography. *FEBS J*, *288*(20), 5788-5804. <https://doi.org/10.1111/febs.15676>

Koehler Leman, J., Mueller, B. K., & Gray, J. J. (2017). Expanding the toolkit for membrane protein modeling in Rosetta. *Bioinformatics*, *33*(5), 754-756. <https://doi.org/10.1093/bioinformatics/btw716>

Kosuri, S., Goodman, D. B., Cambray, G., Mutalik, V. K., Gao, Y., Arkin, A. P.,…Church, G. M. (2013). Composability of regulatory sequences controlling transcription and translation in Escherichia coli. *Proc Natl Acad Sci U S A*, *110*(34), 14024-14029. <https://doi.org/10.1073/pnas.1301301110>

Krivov, G. G., Shapovalov, M. V., & Dunbrack, R. L. (2009). Improved prediction of protein side-chain conformations with SCWRL4. *Proteins*, *77*(4), 778-795. <https://doi.org/10.1002/prot.22488>

Kulp, D. W., Subramaniam, S., Donald, J. E., Hannigan, B. T., Mueller, B. K., Grigoryan, G., & Senes, A. (2012). Structural informatics, modeling, and design with an open-source Molecular Software Library (MSL). *J Comput Chem*, *33*(20), 1645-1661. <https://doi.org/10.1002/jcc.22968>

Lazaridis, T. (2003). Effective energy function for proteins in lipid membranes. *Proteins*, *52*(2), 176-192. <https://doi.org/10.1002/prot.10410>

Liang, B., & Tamm, L. K. (2016). NMR as a tool to investigate the structure, dynamics and function of membrane proteins. *Nat Struct Mol Biol*, *23*(6), 468-474. <https://doi.org/10.1038/nsmb.3226>

Lomize, A. L., & Pogozheva, I. D. (2017). TMDOCK: An Energy-Based Method for Modeling α-Helical Dimers in Membranes. *J Mol Biol*, *429*(3), 390-398. <https://doi.org/10.1016/j.jmb.2016.09.005>

Lomize, M. A., Lomize, A. L., Pogozheva, I. D., & Mosberg, H. I. (2006). OPM: orientations of proteins in membranes database. *Bioinformatics*, *22*(5), 623-625.

MacKerell, A. D., Bashford, D., Bellott, M., Dunbrack, R. L., Evanseck, J. D., Field, M. J.,…Karplus, M. (1998). All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J Phys Chem B*, *102*(18), 3586-3616. <https://doi.org/10.1021/jp973084f>

McKinney, W. (2011). pandas: a Foundational Python Library for Data Analysis and Statistics.

McKinney, W. (2012). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.".

Mirdita, M., Schütze, K., Moriwaki, Y., Heo, L., Ovchinnikov, S., & Steinegger, M. (2022). ColabFold: making protein folding accessible to all. *Nat Methods*, *19*(6), 679-682. <https://doi.org/10.1038/s41592-022-01488-1>

Mougeot, G., Dubos, T., Chausse, F., Péry, E., Graumann, K., Tatout, C.,…Desset, S. (2022). Deep learning -- promises for 3D nuclear imaging: a guide for biologists. *J Cell Sci*, *135*(7). <https://doi.org/10.1242/jcs.258986>

Mueller, B. K., Subramaniam, S., & Senes, A. (2014). A frequent, GxxxG-mediated, transmembrane association motif is optimized for the formation of interhelical Cα-H hydrogen bonds. *Proc Natl Acad Sci U S A*, *111*(10), E888-895. <https://doi.org/10.1073/pnas.1319944111>

Na, D. (2020). User guides for biologists to learn computational methods. *J Microbiol*, *58*(3), 173-175. <https://doi.org/10.1007/s12275-020-9723-1>

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O.,…Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.

Polyansky, A. A., Volynsky, P. E., & Efremov, R. G. (2012). Multistate organization of transmembrane helical protein dimers governed by the host membrane. *J Am Chem Soc*, *134*(35), 14390-14400. <https://doi.org/10.1021/ja303483k>

Roberts, J. B., Nava, A. A., Pearson, A. N., Incha, M. R., Valencia, L. E., Ma, M.,…Keasling, J. D. (2024). Foldy: An open-source web application for interactive protein structure analysis. *PLoS Comput Biol*, *20*(2), e1011171. <https://doi.org/10.1371/journal.pcbi.1011171>

Steinegger, M., & Söding, J. (2017). MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, *35*(11), 1026-1028.

Subramaniam, S., & Senes, A. (2012). An energy-based conformer library for side chain optimization: improved prediction and adjustable sampling. *Proteins*, *80*(9), 2218-2234. <https://doi.org/10.1002/prot.24111>

Tosi, S. (2009). *Matplotlib for Python developers*. Packt Publishing Ltd.

van Iterson, M., van Haagen, H. H., & Goeman, J. J. (2012). Resolving confusion of tongues in statistics and machine learning: a primer for biologists and bioinformaticians. *Proteomics*, *12*(4-5), 543-549. <https://doi.org/10.1002/pmic.201100395>

Vargas, E., Yarov-Yarovoy, V., Khalili-Araghi, F., Catterall, W. A., Klein, M. L., Tarek, M.,…Roux, B. (2012). An emerging consensus on voltage-dependent gating from computational modeling and molecular dynamics simulations. *J Gen Physiol*, *140*(6), 587-594. <https://doi.org/10.1085/jgp.201210873>

Viklund, H., & Elofsson, A. (2008). OCTOPUS: improving topology prediction by two-track ANN-based preference scores and an extended topological grammar. *Bioinformatics*, *24*(15), 1662-1668. <https://doi.org/10.1093/bioinformatics/btn221>

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,…Contributors, S. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods*, *17*(3), 261-272. <https://doi.org/10.1038/s41592-019-0686-2>

Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021.

White, S. H., & Wimley, W. C. (1999). Membrane protein folding and stability: physical principles. *Annu Rev Biophys Biomol Struct*, *28*, 319-365. <https://doi.org/10.1146/annurev.biophys.28.1.319>

Yarov-Yarovoy, V., Baker, D., & Catterall, W. A. (2006). Voltage sensor conformations in the open and closed states in ROSETTA structural models of K(+) channels. *Proc Natl Acad Sci U S A*, *103*(19), 7292-7297. <https://doi.org/10.1073/pnas.0602350103>

Yarov-Yarovoy, V., Schonbrun, J., & Baker, D. (2006). Multipass membrane protein structure prediction using Rosetta. *Proteins*, *62*(4), 1010-1025. <https://doi.org/10.1002/prot.20817>

Zulkower, V., & Rosser, S. (2020). DNA Chisel, a versatile sequence optimizer. *Bioinformatics*, *36*(16), 4508-4509.