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

Computational tools have become more commonly used to advance science forward. With improvements in experimental techniques, high-throughput data allows researchers to more easily parse through and develop conclusions from data using computational algorithms and software developed for analysis. However, developing an understanding for how to use and interpret the data from these high-throughput experiments is difficult without understanding the nuances of how these tools work. As these tools grow, researchers aim to simplify their own understanding to better share these technologies with others (Greener et al., 2022; Mougeot et al., 2022; Na, 2020; van Iterson et al., 2012). In this section, I will describe a variety of computational tools that have been developed to expand the understanding of protein structures from a computational standpoint. I highlight commonly used structure prediction tools such as Rosetta and Alphafold, and then discuss in detail how I developed my protein design algorithm and other programs used to analyze my structures and data.

3.2 Introduction

In 1995, the protein data bank (PDB) was established to collaborate and share protein structures globally online. This tool allows researchers to deposit protein structures after solving them to for others to access 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, researchers focus on using other methods to study MP folding and association, focusing on utilizing a combination of computational tools to estimate the contribution that biophysical forces have on MP stability.

3.3 Studying membrane protein folding and structure

In conjunction with structural determination, computational methods have been invented to evaluate previously solved MP structures. Many methods look to further understand MP folding by establishing energetic terms that aim to estimate the thermodynamics of association. Below I review the current computational methods used to predict protein structures.

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*. 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 algorithm using two energy terms parameterized by CHARMM: hydrogen bonding and van der Waals packing. 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 to assess the extent at which packing can drive MP association. I further detail how I used MSL to design 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. 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.

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

I extracted backbone geometries from all unique MPs found in the Orientations of Membrane Proteins database (OPM) (Lomize et al., 2006). The list of MPs from the PDB website was used to trim for sequence based on 30% sequence similarity, allowing me to search only non-redundant protein structures. Helices of 13AA length are identified and the distances between the Cα carbons are measured. Any two helices with at least 3 Cα carbons within 9 angstroms are extracted as individual protein pairs and the geometric information measured, and the geometry and sequence is stored for analysis. I compiled this geometric data and extracted the angle and xShift, which were plotted against each other for each protein and analyzed using kernel density estimation.

3.4.2 Defining the Interface

A video game screen with a black background

Description automatically generatedIn my first protein design run, I implemented a method to identify the interface of the given geometry using Solvent Accessible Surface Area (SASA). I calculated the SASA for each position on the protein 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 my designs and looking at the results, the variations in the interface increased the complexity of the data (S1). In my future design runs, I standardized the interface for proteins designed in each region through visual inspection of a multitude of geometries (figure). The GASright and right designs used the same interface, while the left designs were composed of 3 unique interfaces to account for known 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 3.3.6.

3.4.3 Developing the energy terms

To assess my designed proteins, I used the same set of energy terms used previously by my lab in CATM. These terms include a term that measures the van der Waals packing (CHARMM\_VDW), hydrogen bonding (SCWRL4\_HBOND), and implicit solvation of the membrane (CHARMM\_IMM1 and CHARMM\_IMM1REF). These terms are then used to assess the energetics of the dimer-monomer during our sequence design search to find the best sequence for the structure. However, computing the monomer energy for each sequence was initially found to create a bottleneck in our design algorithm. To account for this, I developed a BASELINE\_MONOMER term that estimates the monomer energy of each sequence.

A graph with a line graph

Description automatically generated with medium confidence

The BASELINE\_MONOMER term was created by first measuring the energetic contributions from each individual amino acid. These energy contributions were measured as a term that encompasses the following energies: CHARMM\_VDW, SCWRL4\_HBOND, and CHARMM\_IMM1. I computed 1000s of iterations of random 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. 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 of my 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 a subset of amino acids that were used for sequence design and would need to be rerun to establish a term for any additional amino acids. This term was helpful in decreasing computational time.

Another issue that I encountered while designing my proteins was my designed proteins having too many of a single amino acid. This result could impact our protein expression, where designs with many of a single amino acid do not express similarly to designs with a variety of 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: -log(sequence\_probability) x RT (0.592) x weight. First, the number of each AA is counted within the sequence. Then, the number of possible permutations (#permutations) for the set of sequences is determined using the following equation: n!/(#AA1! x #AA2! x …) 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 sequence\_probability is calculated using the frequency of each AA in membrane protein sequences (freq\_AA) to the power of the number of each AA multiplied by the #permutations: (freq\_AA1^#AA1 x freq\_AA2^#AA2 x …) x #permutations.

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 by changing the weights. This allows for versatility in the protein design process. After optimization and testing, the SEQUENCE\_ENTROPY term weight is defaulted to 10, allowing the SEQUENCE\_ENTROPY 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 on the search for the sequence with the energetic minimum. The sequences are saved in an output file to show the trajectory of x sequences that were chosen along with their corresponding energy terms.

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 with max shifts randomly adjusted to a specific value: xShift 0.5, axialRotation 3.0, deltaZ 0.5, and crossingAngle 3.0. The total energy for the altered structure is again used as the acceptance criteria, with the BASELINE\_MONOMER energy being replaced by the calculate 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.

After my protein design for my second run, 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. This program takes in the given structure and repeats the MC based structural refinement, but this time procedurally decreases the 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.

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.

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. These sequences alongside their structures can then be input into the mutation script to identify mutants 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 added 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 better interpret differences between sequences with traditional TOXGREEN values.

3.5.5 Trimming for Clashing Mutations

3.5.6 Hydrogen Bond Mutations

Detail programs and necessary libraries and such here

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>

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>

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.

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>

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.