PepFun 2.0: improved protocols for the analysis of natural and modified peptides

Rodrigo Ochoa

Biophysics of Tropical Diseases, Max Planck Tandem Group, University of Antioquia, 050010 Medellin, Colombia

Email: rodrigo.ochoa@udea.edu.co

Abstract

The role of peptides is nowadays relevant in fields such as drug discovery and biotechnology. Computational analyses are required to study their properties and gain insights into rational design strategies. Both natural and modified peptides containing non-natural amino acids require customized tools to run sequence and structure-based studies. PepFun 2.0 is a new version of the python package for the study of natural and modified peptides using a set of modules to analyze the sequence and structure of the molecules. PepFun 2.0 comprises five main modules for different tasks such as sequence alignments, prediction of properties, generation of conformers, modification of structures, detection of protein-peptide interactions, and extra functions to include peptides containing non-natural amino acids. The code and tutorial are available at: https://github.com/rochoa85/PepFun2

1 Introduction

Motivated by the functional design of peptides as therapeutic agents, several computational packages are currently available to optimize different peptide properties during pre-clinical studies [1, 2, 3]. For example, using tools to run predictions with amino acid scales is standard in analyzing natural

peptides [4, 5]. Additional tools use the peptides' chemical structures to predict conformers and other properties [6, 7]. Given the peptides' hybrid nature, these methods usually combine approaches focused on protein or small molecules but not necessarily peptide-oriented. However, packages like PepFun address this problem through open bioinformatics and cheminformatics protocols to run the analysis with natural peptides in python [8]. The latest can help study different physicochemical properties of peptides and to guide, for example, the generation of analog libraries with potential similar activities [9, 10].

The original PepFun package is a single command line interface to execute peptide analysis tasks, emphasizing sequence and structure functionalities and methods to build peptide libraries based on patterns. However, a modular software architecture and additional functions are required for a broad implementation to study natural and modified peptide sequences composed of non-natural amino acids [11]. Nowadays, including subtle modifications on the peptide sequences is crucial to increase the peptide's structural and metabolic stability, among other applications [12, 13]. For example, being able to run automatic pipelines able to predict conformers, analyze interactions and mutate residues by non-natural monomers is a practice commonly used in both industrial and academic environments [14].

Here PepFun 2.0 is described, a new optimized version of PepFun with functionalities required for researchers designing and studying new natural and modified peptides. Details of the package's five modules, technical assumptions, and best practices are shared. A complete tutorial on implementing the code is available in the repository's README file at: https://github.com/rochoa85/PepFun2.

2 Approach

PepFun 2.0 is written in Python 3, and the code is split into five modules with various functionalities. The package relies on external dependencies, including main packages such as the RDKit (https://rdkit.org/), BioPython (https://biopython.org/) [15], and Modeller [16]. Installing them using a Conda environment through the official package channels is recommended. Additional complementary packages and PepFun 2.0 can be installed using a provided setup file for a quick installation. A set of tests are provided to verify the correct installation. If the user runs functions

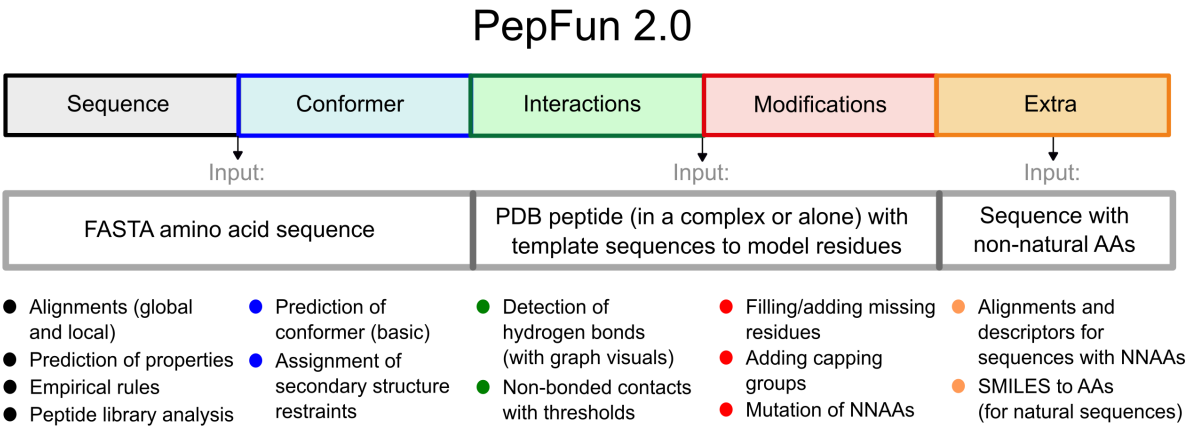

Figure 1: PepFun 2.0 main functionalities. The package is split into five modules: sequence, conformer, interactions, modifications, and extra. Information on the required inputs and the main functionalities are described using a color-code schema. Calling each function from any python script is possible after following the installation instructions.

based on Modeller, an academic or commercial license should be requested from the authors (https://salilab.org/modeller/).

The package contains the scripts: sequence, conformer, modifications, interactions, and extra, which include multiple classes to run analysis such as aligning natural and modified peptide sequences, calculating amino acid and structure-based physicochemical properties, predicting conformers with and without secondary structure restrictions, extract interaction patterns from protein-peptide complexes, as well as modify existing peptide structures by incorporating new residues or adding cap groups. In addition, a method to allow the generation of descriptors for modified peptides is included for training machine learning models. The tasks are summarized in Figure 1, with specific details in the next section.

3 The protocol

PepFun 2.0 can be called from any python script after installing the package. The following are the main scripts and classes.

3.1 Sequence module

The input is a natural peptide sequence in FASTA format. This module contains different methods based on amino acid scales, such as the hydrophobicity Eisenberg scale [17], to calculate properties. One method contains a list of pKa values that depending on the pH, allows the calculation of the peptides' net charge. Other methods are functions from the ProtParam package to calculate metrics such as aromaticity, instability index, and statistics of the amino acid composition [18]. One condition is that the peptide should have natural amino acids.

In addition, the module generates the SMILES representation of the molecule to be subsequently used in packages like the RDKit, allowing the calculation of the peptide molecular weight, lipophilicity by the Crippen LogP [19], and the number of hydrogen bond acceptors and donors. The inclusion of RDKit allows running similarity between pairs of peptides by matching their molecular fingerprints with the Tanimoto coefficient [20]. The alignment can also be done by pairs using a classical position-by-position comparison, which can be weighted using scoring matrices available in the Biopython package. A wrap function to run online blast searches with parameters adjusted for peptide sequences is also available [21].

Finally, the module contains a set of empirical rules to account for solubility and synthesis issues associated with peptides. The rules describe violations by specific patterns or amino acids in the peptide sequence [22]. The higher the number of violations, the lower the probability of validating the peptides experimentally. The solubility rules violations are:

- Warning if the number of charged and/or of hydrophobic amino acids exceeds 45%.
- Warning if the absolute total peptide charge at pH 7 is more than +1.
- Warning if the number of glycines or prolines is more than one in the sequence.
- Warning if the first or the last amino acid is charged.
- Warning if any amino acid represents more than 25% of the total sequence.

The synthesis rules violations are:

- Warning if two prolines are consecutive.
- Warning if the motifs DG (aspartic acid and glycine) and DP (aspartic acid and proline) are present in the sequence. Two rules, one per motif.
- Warning if the sequences ends with aparagine (N) or glutamine (Q) residues.
- Warning if there are charged residues every five amino acids.
- Warning if there are oxidation-sensitive amino acids like methionine (M), cysteine (C) or tryptophan (W). Three rules, one per amino acid.

Overall, the rules allow the filtering of candidates during screening stages, which is crucial to reduce the number of false positives for the prediction of hits.

3.2 Conformer module

The peptide sequence in FASTA format is also required to predict the conformers. For this purpose, two methods are available. One uses RDKit as the engine to generate the structures. Specifically, the sequence is converted to HELM notation and parsed to annotate the amino acids with correct atom names based on the IUPAC nomenclature [23]. Then, the ETKDGv3 method generates the most probable conformer per peptide [24]. The method is a knowledge-based potential optimized for macrocycles that can be useful in the case of small extended peptides with less than 10-15 amino acids [25].

For larger peptides, a second method relies on predicting the peptide's secondary structure. First, the PSIPRED package assigns possible α-helices, β-sheets, or coils [26]. Although the tool focuses on proteins, it can be applied to peptide sequences. However, if the user knows the required secondary structure, it can be provided as a direct input for the Modeller pipeline. Second, a loop refinement protocol with Modeller uses one single amino acid of the peptide as a template (i.e., a predicted amino acid structure using RDKit). Then the rest of the residues are modeled following the secondary structure restraints. If the peptide is cyclic, the function can add restrictions on the formation of disulfide bonds.

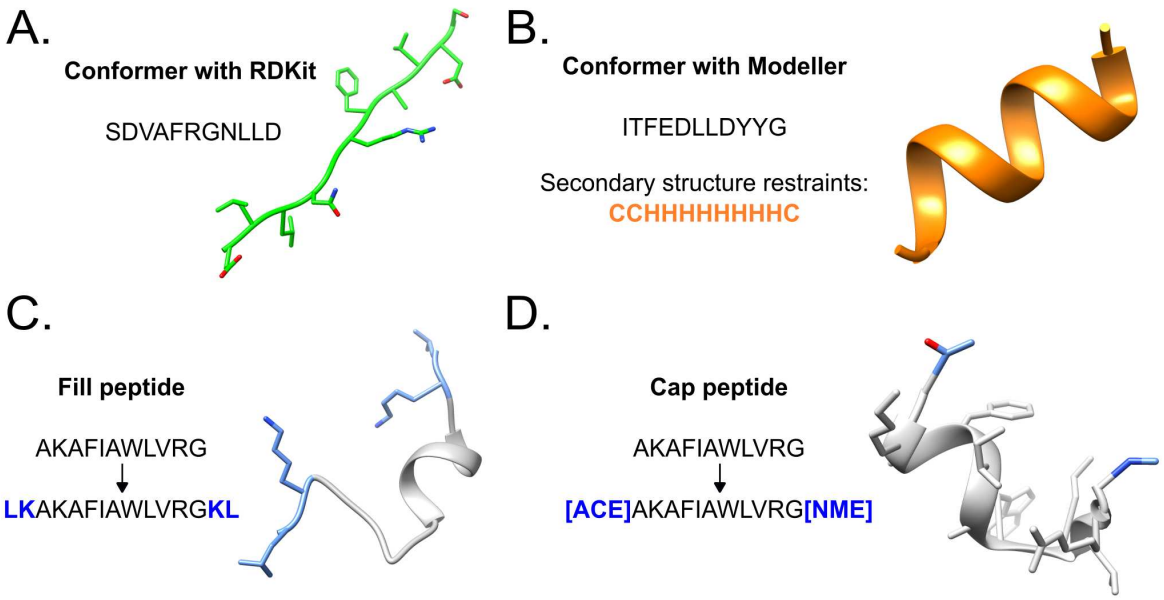

Figure 2: Examples of the Conformer and Modifications modules. (A) Prediction of an extended conformer for the peptide sequence SDVAFRGNLLD using the ETKDGv3 module from RDKit [24]. (B) Prediction of a helical conformer for peptide ITFEDLLDYYG using a combination of PSIPRED [26] and Modeller [16] to predict the secondary structure and generate the 3D model using the secondary structure restraints in a loop refinement protocol. (C) Filling a peptide by adding two amino acids, LK at the N-terminal and KL at the C-terminal part, using the sequence AKAFIAWLVRG. (D) Capping the peptide AKAFIAWLVRG using the acetyl group (ACE) at the N-terminal and methylamine (NME) at the C-terminal part.

Figure 2 shows an example of conformers predicted with the two methodologies. In Figure 2A, an extended conformation is predicted for an 11-mer peptide using the RDKit protocol. In Figure 2B, the peptide is predicted as a helical conformer, which is consistent with the experimental folding of the bound peptide (PDB id 2buo) ([27]).

3.3 Modifications module

This module's purpose is to modify existing structures of peptides by including missing amino acids, capping groups, or by mutation of specific residues. The input is a PDB file of the peptide alone or

in a complex with a protein target. Something important is assigning the correct atom names and numbering in the input file to facilitate its manipulation using the BioPython and Modeller packages.

Using Modeller, the user can provide a PDB file of the peptide alone or in a complex for the filling functionalities. The module also receives the amino acid sequence and the sequence of a new peptide with new residues at different positions. The Modeller pipeline can also modify the peptide when bound to a protein target. For the capping functionality, it is possible to add glycines at any flanking parts using Modeller and then convert them to an acetyl group at the N-terminal or methylamine at the C-terminal part. An example of the filling function is shown in Figure 2C, where the two residues at each flanking part are added to the existing peptide PDB structure. Figure 2D describes an example of capping a peptide at both terminal parts.

Finally, a mutation class is added to the module to mutate a natural amino acid by a non-natural amino acid (NNAA) of interest. This class is one method that allows the analysis of non-natural residues in this version of PepFun. The method is inspired in the PeptideBuilder package [28] by using BioPython as the basis to manipulate the structures, calculate parameters, and assign new atoms in the NNAA side chain. To run the tool, a PDB file of the NNAA with the coordinates and correct atom naming is necessary. The file can be obtained from the PDB or predicted using conformer generators. The mutated peptide will incorporate the adapted coordinates of the NNAA, which can be used as input for simulations if the force field parameters are available.

3.4 Interactions module

A protein-peptide complex in PDB format is required in the interactions module as input. The module contains different functions. The first one runs the DSSP program to assign secondary structure elements to each residue in the peptide [29]. The second one uses the same DSSP program to detect the possible hydrogen bonds between the peptide and the protein. A detailed report is generated with the detected hydrogen bonds per residue/atom and a graph-based figure with the interactions, where the nodes are the residues and the edges interactions. The nodes are colored based on the chain, and the width of the interactions represents if more than one potential hydrogen bond is present. For this module, it is possible to have NNAAs in the peptide.

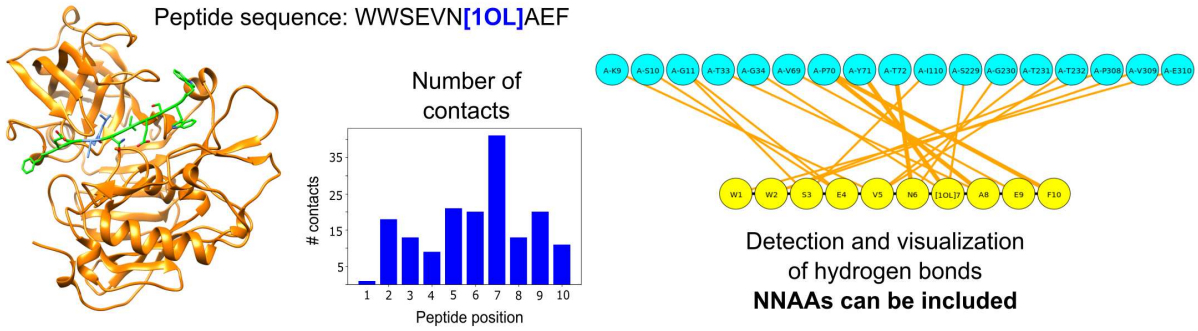

Figure 3: Interactions detected for a protein-peptide complex with PDB id 1xn2 [30]. The peptide sequence (shown in green in the structure) is WWSEVN[1OL]AEF, where 1OL is a non-natural amino acid (NNAA) available in the PDB. The hydrogen bond graph recognizes the interactions between the protein (shown in orange in the structure) as edges between nodes, which have different colors if they are part of the protein or the peptide. A plot with the number of contacts per residue using a threshold of 4 Å is also shown.

The module also allows the detection of hydrophobic contacts. For that goal, a function based on the Biopython package detects all the possible distances between the protein and the peptide residues under a defined threshold, which will depend on the user's requirements. By default, a threshold of 4 Å is used. An example of the protein-peptide complex with PDB id 1xn2 [30] is shown in Figure 3. The graph-based representation is generated using the igraph module from python [31].

3.5 Extra module

The extra module includes specialized functions to analyze modified peptides with NNAAs. The inputs for the modified peptides are sequences using the BILN notation [32]. In this notation, the monomers are separated by dashes, and the NNAAs are described based on codes reported in official monomer repositories. PepFun 2.0 includes the open HELM monomer database to map 322 non-natural elements (https://github.com/PistoiaHELM/HELMMonomerSets).

The first method is the alignment and score of modified peptides using the pairwise2 functionality of Biopython. The alignment can be unweighted (i.e., matches and mismatches) or weighted by a similarity-based scoring matrix generated between all the monomers. A function to update the matrix

using any similarity threshold is also included. The formation of gaps is allowed in the alignment, and the score denotes the similarity between the peptides.

A second method is the generation of descriptors for modified peptides based on pre-calculated properties of the HELM monomer dataset. These properties are molecular weight, topological polar surface area (TPSA), partition coefficient (LogP), and the number of rotatable bonds. Then a function generates autocorrelation of these properties using the Moran equation [33], described as:

$$M(d) = \frac{\frac{1}{N-d}\sum_{i=1}^{N-d}(P'_i - \bar{P}'_t)(P'_{i+d} - \bar{P}_t)}{\frac{1}{N}\sum_{i=1}^{N}(P'_i - \bar{P}_t)^2} \quad d = 1,2,3,...,n \tag{1}$$

where each property P'_i is normalized according to:

$$P'_i = \frac{P_i - \bar{P}}{\sigma} \tag{2}$$

P_i is the original property value, P_t is the average of all the amino acids, $\bar{P}$ is the mean value, and σ is the standard deviation calculated based on the property values of the available amino acids. The number of descriptors will depend on the variable d, which is adjusted to the peptide sequence length. A graphical example of the alignment of modified peptides and the autocorrelation of properties to generate amino acid-based descriptors is shown in Figure 4

The last method allows converting the SMILES notation of a peptide to its FASTA sequence. A function maps natural amino acids by detecting the masses of the single residues based on RDKit calculations. The input can be adapted to accept standard input formats like SDF files.

4 Conclusion

PepFun 2.0 is a package that can be installed under a Linux environment and called through its independent modules in a python script. The functionalities included are motivated by everyday tasks required to study and design new peptide variants, with the possibility now to include modified

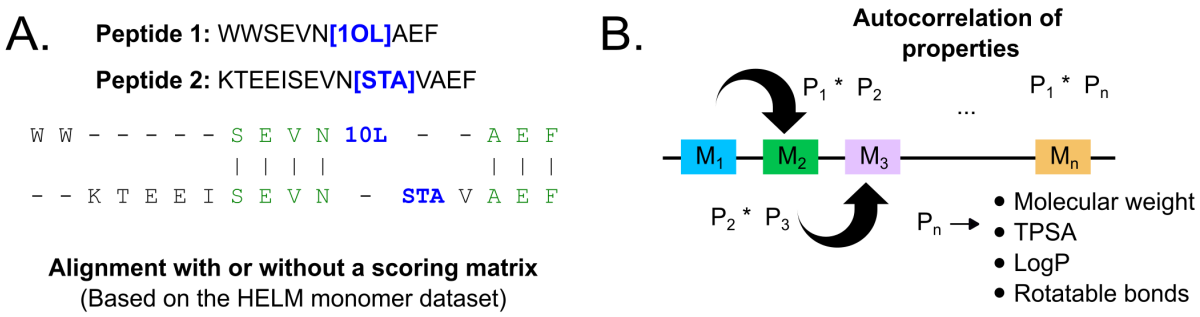

Figure 4: Examples of analysis for modified peptides with non-natural amino acids (NNAA). (A) Alignment of two peptide sequences having each a single NNAA (in blue). The alignment allows the generation of gaps to fit the matches. (B) Schema of the Moran equation to generate amino acid-based descriptors based on the autocorrelation of properties per monomer. The properties are the molecular weight, topological polar surface area (TPSA), partition coefficient (LogP), and the number of rotatable bonds.

peptides composed of non-natural amino acids available in open repositories. The package is open for academic purposes, with examples of how to run each module in the code repository.

Availability

- Project name: PepFun (version 2.0)
- Project home page: https://github.com/rochoa85/PepFun2
- Operating system(s): Linux
- Programming language: Python 3
- Other requirements: RDKit 2020 or higher; Biopython 1.7.9 or higher; Modeller 10.3 or higher.
- License: MIT

The code is available as a Github repository. Any questions related with the implementation can be directed to the author's email accounts.

Acknowledgements

I thank Dr. Pilar Cossio for valuable insights into the conceptualization of PepFun. I also thank Dr. Thomas Fox for the feedback on some of the package's tools.

Funding

This work has been supported by MinCiencias, University of Antioquia and Ruta N, Colombia, the Max Planck Society, Germany.

Conflict of interest

The author declare there are no competing interest.

References

[1] D. S. Paul and N. Gautham, "Mols 2.0: software package for peptide modeling and protein–ligand docking," *Journal of molecular modeling*, vol. 22, no. 10, pp. 1–9, 2016.

[2] J. L. Baylon, O. Ursu, A. Muzdalo, A. M. Wassermann, G. L. Adams, M. Spale, P. Mejzlik, A. Gromek, V. Pisarenko, D. Hancharyk, *et al.*, "Pepsea: Peptide sequence alignment and visualization tools to enable lead optimization," *Journal of Chemical Information and Modeling*, vol. 62, no. 5, pp. 1259–1267, 2022.

[3] R. Ochoa, M. A. Soler, A. Laio, and P. Cossio, "Parce: Protocol for amino acid refinement through computational evolution," *Computer Physics Communications*, vol. 260, p. 107716, 2021.

[4] S. Duvaud, C. Gabella, F. Lisacek, H. Stockinger, V. Ioannidis, and C. Durinx, "Expasy, the swiss bioinformatics resource portal, as designed by its users," *Nucleic Acids Research*, vol. 49, no. W1, pp. W216–W227, 2021.

[5] S. Kawashima and M. Kanehisa, "Aaindex: amino acid index database," *Nucleic acids research*, vol. 28, no. 1, pp. 374–374, 2000.

[6] Y. Shen, J. Maupetit, P. Derreumaux, and P. Tufféry, "Improved pep-fold approach for peptide and miniprotein structure prediction," *Journal of chemical theory and computation*, vol. 10, no. 10, pp. 4745–4758, 2014.

[7] Y. Yan, D. Zhang, and S.-Y. Huang, "Efficient conformational ensemble generation of protein-bound peptides," *Journal of cheminformatics*, vol. 9, no. 1, pp. 1–13, 2017.

[8] R. Ochoa and P. Cossio, "Pepfun: open source protocols for peptide-related computational analysis," *Molecules*, vol. 26, no. 6, p. 1664, 2021.

[9] M. Tu, S. Cheng, W. Lu, and M. Du, "Advancement and prospects of bioinformatics analysis for studying bioactive peptides from food-derived protein: Sequence, structure, and functions," *TrAC Trends in Analytical Chemistry*, vol. 105, pp. 7–17, 2018.

[10] J. Joshi and D. Blankenberg, "Pdaug: a galaxy based toolset for peptide library analysis, visualization, and machine learning modeling," *BMC bioinformatics*, vol. 23, no. 1, pp. 1–17, 2022.

[11] K. N. Amarasinghe, L. De Maria, C. Tyrchan, L. A. Eriksson, J. Sadowski, and D. Petrovic, "Virtual screening expands the non-natural amino acid palette for peptide optimization," *Journal of Chemical Information and Modeling*, vol. 62, no. 12, pp. 2999–3007, 2022.

[12] T. Zimmermann, L. Thomas, T. Baader-Pagler, P. Haebel, E. Simon, W. Reindl, B. Bajrami, W. Rist, I. Uphues, D. J. Drucker, *et al.*, "Bi 456906: Discovery and preclinical pharmacology of a novel gcgr/glp-1r dual agonist with robust anti-obesity efficacy," *Molecular Metabolism*, vol. 66, p. 101633, 2022.

[13] D. Gfeller, O. Michielin, and V. Zoete, "Swisssidechain: a molecular and structural database of non-natural sidechains," *Nucleic acids research*, vol. 41, no. D1, pp. D327–D332, 2012.

[14] R. Ochoa, P. Cossio, and T. Fox, "Protocol for iterative optimization of modified peptides bound to protein targets," *Journal of Computer-Aided Molecular Design*, vol. 36, no. 11, pp. 825–835, 2022.

[15] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, *et al.*, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[16] N. Eswar, D. Eramian, B. Webb, M.-Y. Shen, and A. Sali, "Protein structure modeling with modeller," in *Structural proteomics*, pp. 145–159, Springer, 2008.

[17] D. Eisenberg, R. M. Weiss, and T. C. Terwilliger, "The hydrophobic moment detects periodicity in protein hydrophobicity.," *Proceedings of the National Academy of Sciences*, vol. 81, no. 1, pp. 140–144, 1984.

[18] P. Artimo, M. Jonnalagedda, K. Arnold, D. Baratin, G. Csardi, E. De Castro, S. Duvaud, V. Flegel, A. Fortier, E. Gasteiger, *et al.*, "Expasy: Sib bioinformatics resource portal," *Nucleic acids research*, vol. 40, no. W1, pp. W597–W603, 2012.

[19] R. Mannhold and H. Van de Waterbeemd, "Substructure and whole molecule approaches for calculating log p," *Journal of Computer-Aided Molecular Design*, vol. 15, no. 4, pp. 337–354, 2001.

[20] D. R. Flower, "On the properties of bit string-based measures of chemical similarity," *Journal of chemical information and computer sciences*, vol. 38, no. 3, pp. 379–386, 1998.

[21] J. Ye, S. McGinnis, and T. L. Madden, "Blast: improvements for better sequence analysis," *Nucleic acids research*, vol. 34, no. suppl_2, pp. W6–W9, 2006.

[22] G. B. Santos, A. Ganesan, and F. S. Emery, "Oral administration of peptide-based drugs: beyond lipinski's rule," *ChemMedChem*, vol. 11, no. 20, pp. 2245–2251, 2016.

[23] J. Milton, T. Zhang, C. Bellamy, E. Swayze, C. Hart, M. Weisser, S. Hecht, and S. Rotstein, "Helm software for biopolymers," *Journal of Chemical Information and Modeling*, vol. 57, no. 6, pp. 1233–1239, 2017.

[24] S. Riniker and G. A. Landrum, "Better informed distance geometry: using what we know to improve conformation generation," *Journal of chemical information and modeling*, vol. 55, no. 12, pp. 2562–2574, 2015.

[25] S. Wang, K. Krummenacher, G. A. Landrum, B. D. Sellers, P. Di Lello, S. J. Robinson, B. Martin, J. K. Holden, J. Y. Tom, A. C. Murthy, *et al.*, "Incorporating noe-derived distances in conformer generation of cyclic peptides with distance geometry," *Journal of Chemical Information and Modeling*, vol. 62, no. 3, pp. 472–485, 2022.

[26] D. W. Buchan and D. T. Jones, "The psipred protein analysis workbench: 20 years on," *Nucleic acids research*, vol. 47, no. W1, pp. W402–W407, 2019.

[27] F. Ternois, J. Sticht, S. Duquerroy, H.-G. Kräusslich, and F. A. Rey, "The hiv-1 capsid protein c-terminal domain in complex with a virus assembly inhibitor," *Nature structural & molecular biology*, vol. 12, no. 8, pp. 678–682, 2005.

[28] M. Z. Tien, D. K. Sydykova, A. G. Meyer, and C. O. Wilke, "Peptidebuilder: A simple python library to generate model peptides," *PeerJ*, vol. 1, p. e80, 2013.

[29] W. Kabsch and C. Sander, "Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features," *Biopolymers: Original Research on Biomolecules*, vol. 22, no. 12, pp. 2577–2637, 1983.

[30] R. T. Turner, L. Hong, G. Koelsch, A. K. Ghosh, and J. Tang, "Structural locations and functional roles of new subsites s5, s6, and s7 in memapsin 2 (β-secretase)," *Biochemistry*, vol. 44, no. 1, pp. 105–112, 2005.

[31] G. Csardi, T. Nepusz, *et al.*, "The igraph software package for complex network research," *InterJournal, complex systems*, vol. 1695, no. 5, pp. 1–9, 2006.

[32] T. Fox, M. Bieler, P. Haebel, R. Ochoa, S. Peters, and A. Weber, "Biln: A human-readable line notation for complex peptides," *Journal of Chemical Information and Modeling*, vol. 62, no. 17, pp. 3942–3947, 2022.

[33] Z. Chen, P. Zhao, F. Li, A. Leier, T. T. Marquez-Lago, Y. Wang, G. I. Webb, A. I. Smith, R. J. Daly, K.-C. Chou, *et al.*, "ifeature: a python package and web server for features extraction and selection from protein and peptide sequences," *Bioinformatics*, vol. 34, no. 14, pp. 2499–2502, 2018.