# Protein theory part

January 23, 2020

## 1. How do these methods deal differently with local versus non-local structure?

**AlphaFold.** For modeling the non local structure, AlphaFold uses a convolutional neural network that is trained on PDB structures to predict the distances between the $C_\beta$ atoms of pairs, $i, j$, of residues of a protein. Given an amino acid sequence, $S$, it derive features from an MSA($S$) of that sequence and the networks predict the probability of resideus being separated by different distances. The network is trained to predict distances between two 64-residue fragments of a chain, than they combine these discrete probability disributions to produce distance predictions for the entire protein (distogram) [1].

In order to model the local structure, they use a convolutional auto-regressive latent neural network based on the work of Karol Gregor on recurrent neural network for image generation [2]. Basically a separate output of the contact prediction network is trained to predict discrete probability distributions of backbone torsion angles $P\left(\varphi_i, \psi_i | S, \mathrm{MSA}(S)\right)$, and after fitting a von Mises distribution, this is used to add a smooth torsion modelling term to the potential. Finally a Rosetta2 score is added to the potential to prevent steric clashes [1].

Than the combined potential (torsion potential, Rosetta2 score and the log probability of the distance predictions) is optimized by gradient descent to achieve accurate structure predictions [3].

**RGN.** Recurrent Geometric Networks (RGN) couples local and global protein structure via geometric units that optimize global geometry without violating local covalent chemistry. Protein sequences are fed one residue at a time to the computational units of an RGN, which compute an internal state that is integrated with the states of adjacent units. Based on these computations, torsional angles are predicted and fed to geometric units, which sequentially translate them into Cartesian coordinates to generate the predicted structure. Finally RGN parameters are optimized to minimize the dRMSD between predicted and experimental structures using backpropagation [4].

## 2.1. How is a "reference state" used by AlphaFold and why?

AlphaFold use a reference state to obtains the correct distance-based potential. The distance potential is created from the negative log likelihood of the distances, summed over all pairs of residues $i, j$.

$$V_{\text{distance}}\left(\mathbf{x}\right) = -\sum_{i,j,i\neq j} \log P\left(d_{ij} | S, \mathrm{MSA}(S)\right) \tag{1}$$

Given a protein with N residues, this potential accumulates $L^2$ terms from marginal distribution predictions, so the reference state is used to correct the overrepresentation of the prior by subtracting the reference distribution from the distance potential in the log domain. The distance potential

with a reference state become:

$$V_{\text{distance}}\left(\mathrm{x}\right) = -\sum_{i,j,i\neq j} \log P\left(d_{ij}|\mathcal{S}, \mathrm{MSA}(\mathcal{S})\right) - \log P\left(d_{ij}|\,\text{length}, \delta_{\alpha\beta}\right) \tag{2}$$

The reference distribution used by AlphaFold models the distance distributions independent of the protein sequence and is computed by training a small version of the distance prediction neural network on the same structures, without sequence or MSA input features [1].

### 2.2. What is the purpose of the reference state?

The prediction of protein structure requires conformational-energy-based score functions that can correctly pick the native conformation out of a large number of incorrect folds. In order to properly evaluate the nativeness of the interactions in a given conformation, its conformational energy is measured relative to a so-called reference state, a hypothetical "random" state where those interactions are absent [5].

### 2.3. Why does the end-to-end prediction not use a reference state?

RGN doensn't need a reference state because it doesn't use any sampling. His neural network makes a local prediction of angles and then "remember" it to makes a global prediction so there is no need for bias correction [4].

## 3. To what extend are they knowledge-based and to what extend are they physics-based?

**AlphaFold.** AlphaFold uses a combined potential of the mean force that can accurately describe the shape of a protein, and than it optimizes it by a simple gradient descent algorithm. Its protein-specific potential combines the distance potential, the torsional potential and the Rosetta2 score:

$$V_{\text{total}}\left(\phi,\psi\right) = V_{\text{distance}}\left(G(\phi,\psi)\right) + V_{\text{torsion}}\left(\phi,\psi\right) + V_{\text{score2 smooth}}\left(G(\phi,\psi)\right) \tag{3}$$

The distance potential and the torsional potential represent the kowledge based potentials, they are used to model the non local and local structure respectively. Finally Rosetta2 score, which include a van der Waals term, is the physics based potential used to prevent steric clashes [1].

**RGN.** RGN energy potential is entirely knowledge based because it doesn't use any physics based energy function. That is the main reason why RGN-predicted structures having non-physical torsion angles can often have a poor local structure.[6]

## 4. What are their major similarities and major differences?

They both are ab initio modeling methods that use neural networks for protein structure prediction [3]. One of the main difference is that AlphaFold is a co-evolutionary method that convert its predictions into geometric constraints to guide a conformation sampling process [4]. It can predict distances with sufficient accuracy to outperform state-of-the-art search methods, but because of the complexity of this method the prediction can still be slow taking tens to hundreds of hours [7]. In contrast a trained RGN model, which doesn't use evolutionary data, is a single mathematical function evaluated once per prediction, and while training RGNs can take weeks to months, once trained, they are able to make predictions in few seconds [4]. An other difference, as already

mentioned, is in how they deal with local versus non-local structure and in the energy potential they use. Because of these differences, AlphaFold model local structure much better than long-range interaction [3], while RGN often predict global fold correctly but do less well with secondary structure [4].

## 5. In your opinion, what are their strengths and limitations compared to each other?

One of the greatest strenght of AlphaFold is probably the level of accuracy that its predictions can achieve, while one of the greatest RGN strength is probably the speed of its predictions. The main weakness of AlphaFold approach is that it still relies heavily on coevolution [3]. Because coevolutionary methods do not construct a model of the relationship between individual sequences and structures, they are unable to predict structures for which no sequence homologs exist. Also, such methods are generally unable to predict the structural consequences of minor sequence changes such as mutations, because they operate on protein families rather than individual sequences [4]. One limitation of current RGNs is their reliance on PSSMs. PSSMs are much weaker than multiple sequence alignments, as they are based on single residue mutation frequencies and ignore how each residue mutates in response to all other residues [4].

## References


[1] Andrew W. Senior, Richard Evans et al. (2020) Improved protein structure prediction using potentials from deep learning. Nature.

[2] Karol Gregor et al. (2015) DRAW: A Recurrent Neural Network For Image Generation

[3] Andrew W. Senior, Richard Evans et al. (2019) Protein structure prediction using multiple deep neuralnetworks in the 13th Critical Assessment of ProteinStructure Prediction (CASP13). Proteins.

[4] Mohammed AlQuraishi. (2019) End-to-End differentiable learning of protein structure. Cell Systems 8, 292–301.

[5] Armando D Solis and Shalom R Rackovsky. (2011) Information-Theoretic Analysis of the Reference State in Contact Potentials used for Protein Structure Prediction. Proteins.

[6] AlQuraishi, M. (2020). AlphaFold @ CASP13: "What just happened?". Some Thoughts on a Mysterious Universe.

[7] Mohammed AlQuraishi. (2020) Protein-structure prediction gets real. Nature.

# Protein practical part

January 23, 2020

## 1. Introduction

The task of this exercise is to investigate the distributions of RMSD scores between side chains pairs of 18 different amino acids. The two side chains from each pair come from different proteins, this is a way to compare the variability of the amino acid structures and their difference. Gly and Ala are excluded from the analysis since their side chains are too small, and without enough degrees of freedom, to present a significative difference in terms of structural variability.

## 2. Materials and methods

For the exercise we used the Top500 database of PDB files, available from Richardson Lab Web Site. Richardson and collegues, from Duke University, used this data for their Ramachandran and rotamer studies. This is a selection of 500 files from the Protein Data Bank (PDB) that are high resolution (1.8 Å or better), low homology, and high quality [1]. The PDB format provides a standard representation for macromolecular structure data derived from X-ray diffraction and NMR studies [2].

The programming language we used to perform the analysis is Python 3. In addition we used NumPy package to do operations with vectors and matrices, Matplotlib to plot the histograms and Bio.Python to work with the PDB files. In particular we used a Bio.Python module called Bio.PDB, the module has been developed by Thomas Hamelryck and focuses on working with crystal structures of biological macromolecules. It contains a parser for PDB files that makes the atomic information available in an easy-to-use but powerful data structure [3].

### 2.1. Implementation

For my implementation I used five funcions that I will not completely report here to avoid redundancy. The first function extract the protein structures from a given directory. The second function extract the atoms coordinates of the side chain of a given residue, calculate the side chain center of mass and return the centered set of coordinates. The third function superimpose two residues (side chains), represented by two 3 by n numpy matrices, and return their minimum RMSD. The fourth function, used in combination with the others, extract 1000 side chains pairs randomly sampled from the protein data set. The last function is used to make and save the plots of the RMSD distributions.

**Parsing the structure.** I start my implementation by parsing all the structure contained in the Top500H directory, try and except are used in order to ignore the structure that can not be parsed by the PDBParser. I use the miscellaneous operating system interfaces (OS) module to access the PDB files contained in the directory.

```
[ ]: p = PDBParser(QUIET = True)
     list_structures = []
for filename in os.listdir(directory):
try:
     s = p.get_structure(filename, os.path.join(directory, filename))
     list_structures.append(s)
except:
     print(filename, "can't be parsed")
```

**Extract side chain pairs.** I select randomly two protein structures using random.choice() from NumPy, than I extract all the selected amino acids from each protein and I choose randomly the pair of amino acids.

```
[ ]: # Select two random different proteins
i,j = random.choice(len(list_structures), size = 2, replace = False)
s1 = list_structures[i]
s2 = list_structures[j]
# Extract all the selected amino acid from the two selected proteins
list_res1 = []
for res in s1[0].get_residues():
    if res.get_resname() == aa:
        list_res1.append(res)
list_res2 = []
for res in s2[0].get_residues():
    if res.get_resname() == aa:
        list_res2.append(res)
```

Than if both proteins contains the selected amino acid, I obtain the centered coordinates of their side chains (method described in the next subsection). At this point if the two side chains have the same number of atoms I compute the RMSD score.

```
[ ]: # Check if both proteins have the selected amino acid
if len(list_res1) != 0 and len(list_res2) != 0:
    # Select randomly one amino acid from each of the two proteins
    i1 = random.choice(len(list_res1))
    i2 = random.choice(len(list_res2))
    res1 = list_res1[i1]
    res2 = list_res2[i2]
    # Get the centered coordinates of the two side_chains
    sc1 = get_centered_sidechain(res1)
    sc2 = get_centered_sidechain(res2)
    # Check if the side chains have the same number of atoms
    if sc1.shape == sc2.shape:
        # Calculate RMSD and append it to the RMSD list
        rmsd = calc_RMSD(sc1, sc2)
        rmsd_list.append(rmsd)
```

**Optimal RMSD superposition.** The implementation of the RMSD algorithm (and most of the rest of the code) is based on weekly exercises solutions provided by our Structural Bioinfomatics professor Thomas Hamelryck [4].

In order to measure the structural similarity between side chain pairs, we used the root-mean-square deviation (RMSD) of atomic positions, which is simply the square root of the distance between all atoms divided by their number. We want to apply a U rotation matrix to y, until the RMSD is minimized.

$$\mathrm{RMSD}(\boldsymbol{x}, \boldsymbol{y}) = \min_{U} \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} |x_i - U y_i|^2} \tag{1}$$

In the exercise the centers of mass of the two sets of vectors used for the RMSD calculation are not at their origin, so I centered the atoms before applying the optimal RMSD superposition. Since the task was to compare the structural similarities between side chains of the same amino acid, I calculated the center of mass (COM) by adding all the coordinates of the side chain atoms to a vector, including the alpha carbon and excluding all hydrogen. Than I divided that vector for the number of atoms (N).

$$COM = \frac{1}{N} \sum_{i=1}^{N} (a_{ix}, b_{iy}, c_{iz}) \tag{2}$$

Finally I centered the atoms by subtracting the center of mass to each coordinate vector in the set. Once I obtained the centered coordinates I wanted to find the rotation matrix U that minimize the RMSD score, so I applyed the singular value decomposition (SVD) to the correlation matrix R.

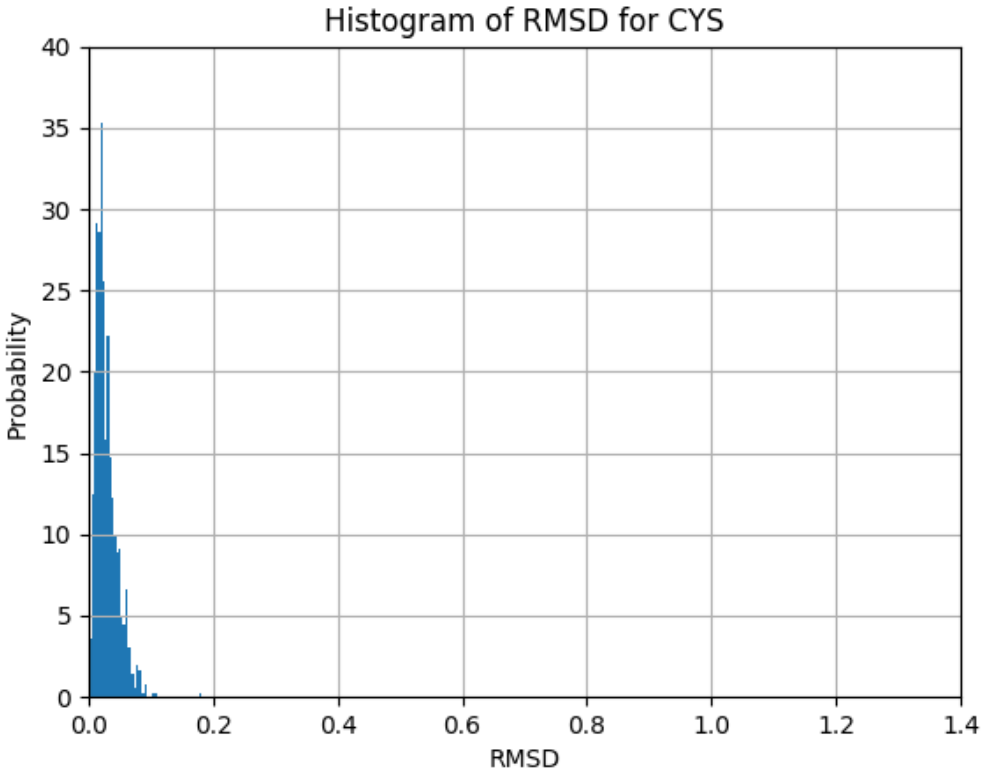
$$R = YX^t = VSW^t \tag{3}$$

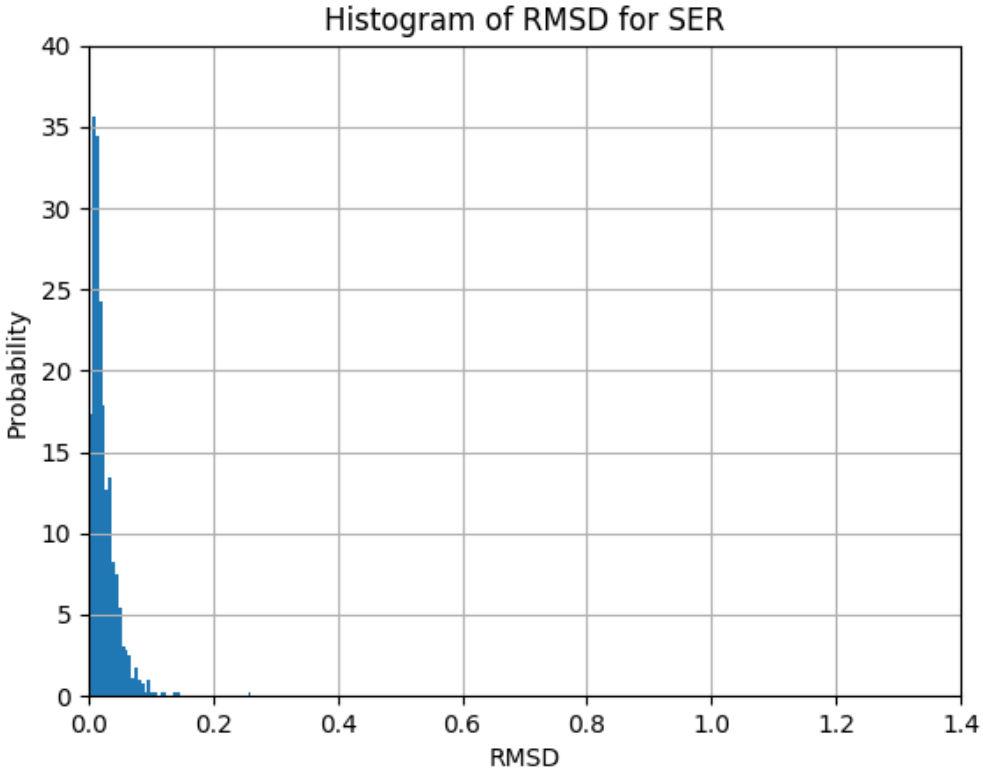
$$U_{\min} = WV^t \tag{4}$$

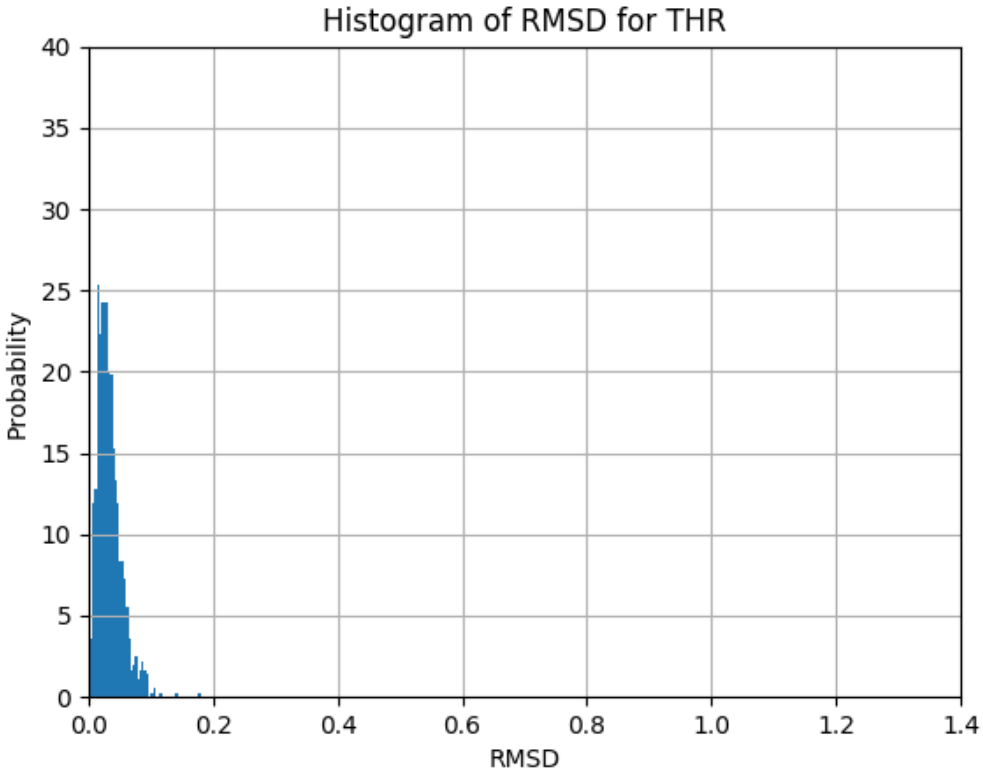
Sometimes the rotation matrix U that minimize the RMSD is a roto-inversion, that will superimpose a mirror image. To avoid that we have to multiply the components of the rotation matrix U for $Z = diag(1, 1, -1)$, and we also change the sign to the third element of the diagonal matrix S (even that this is not necessary in the calculation of the RMSD from the coordinates). Than I applyed the rotation matrix U to y and I finally calculated the RMSD from the two set of coordinates.

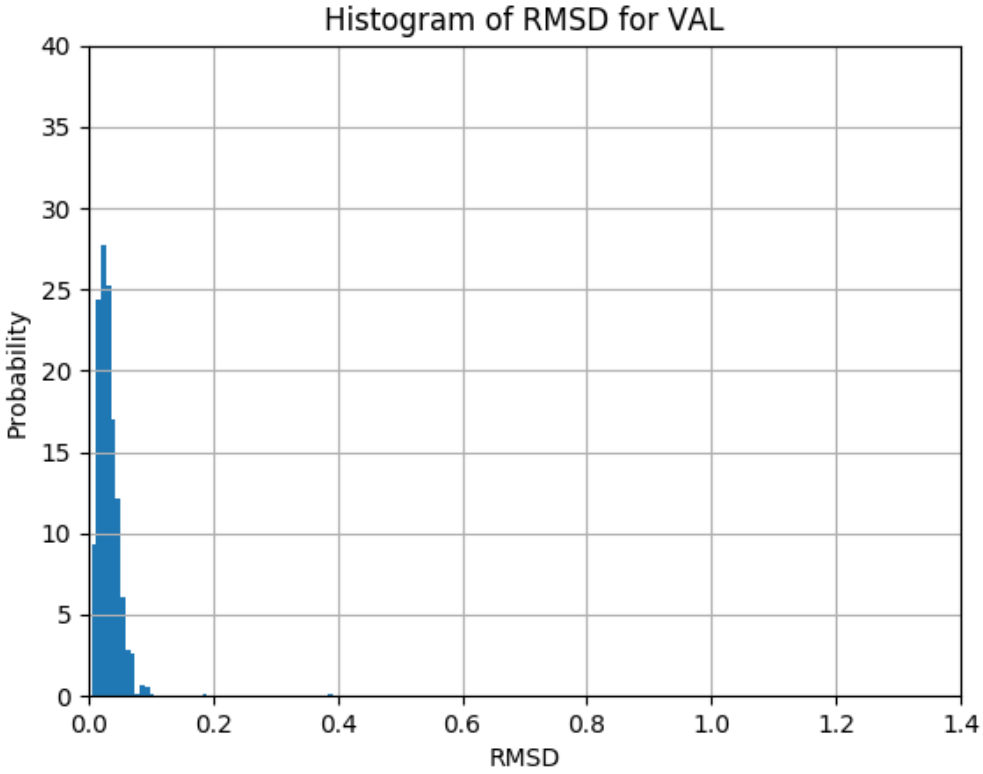
## 3. Results

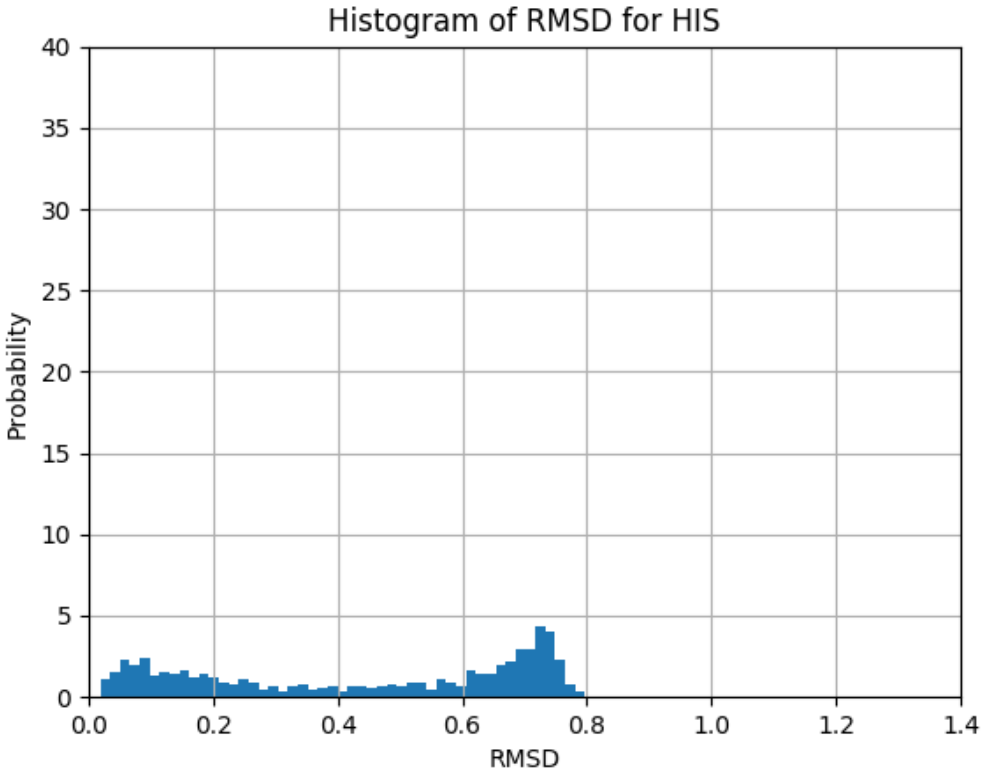
The final results of the exercise are the plots of the RMSD scores distributions of the 18 different amino acids. It is possible to observe that, with some exceptions (ASN and ASP), they show three main distributions shapes.

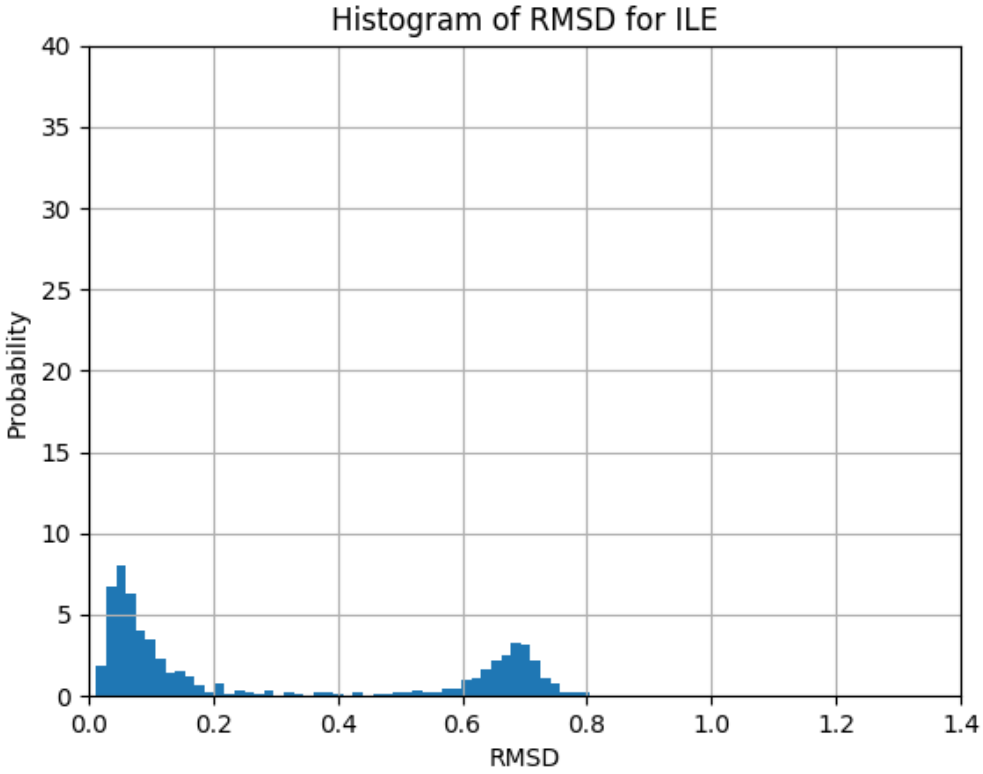

Histogram of RMSD for CYS
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

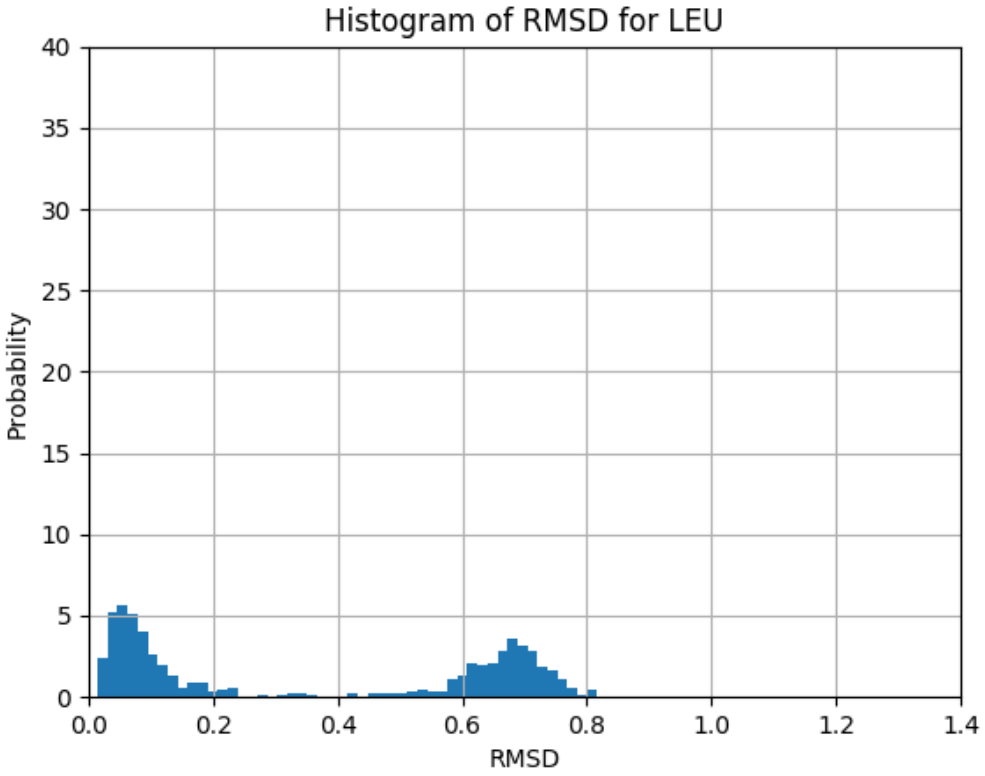

Histogram of RMSD for SER
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

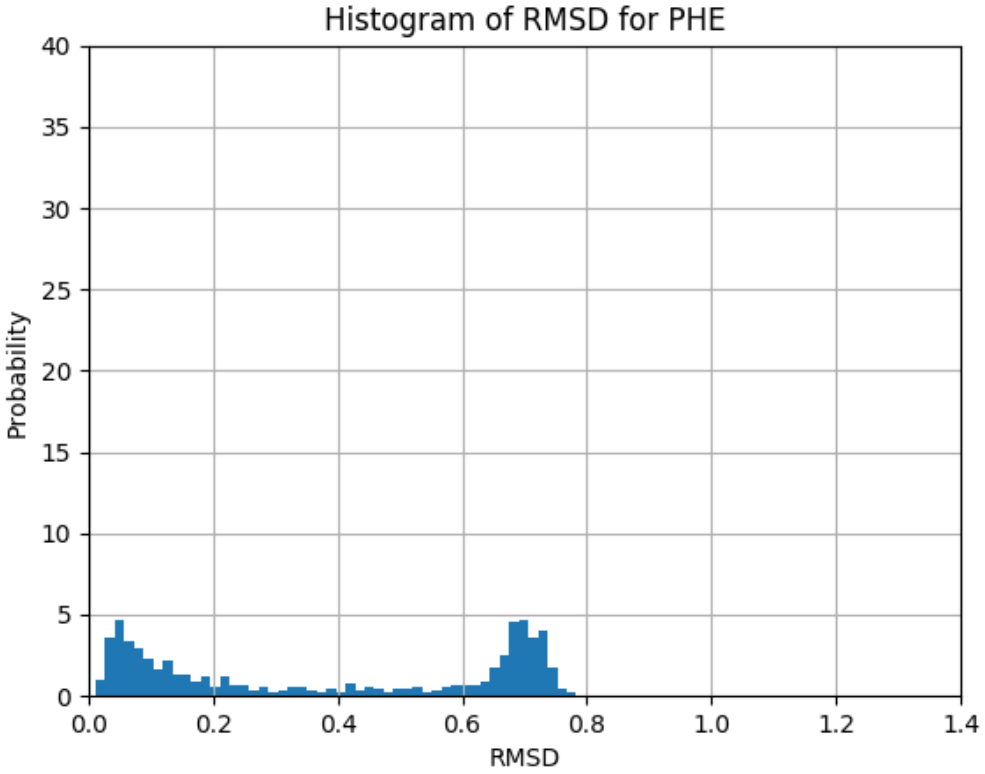

Histogram of RMSD for THR
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

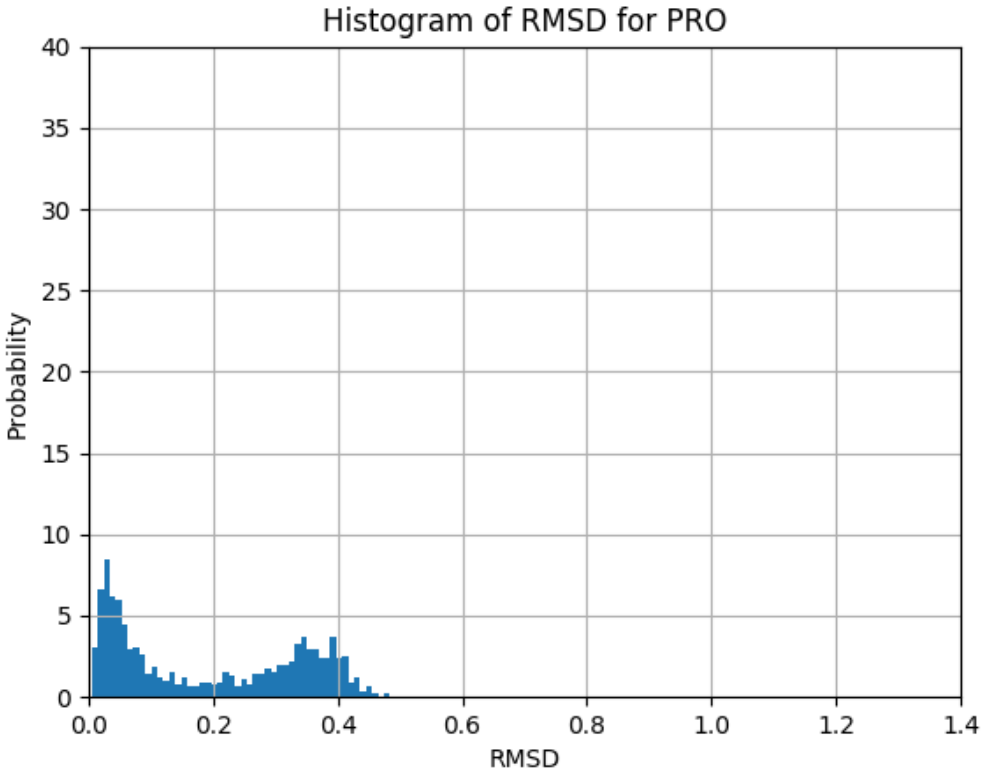

Histogram of RMSD for VAL
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

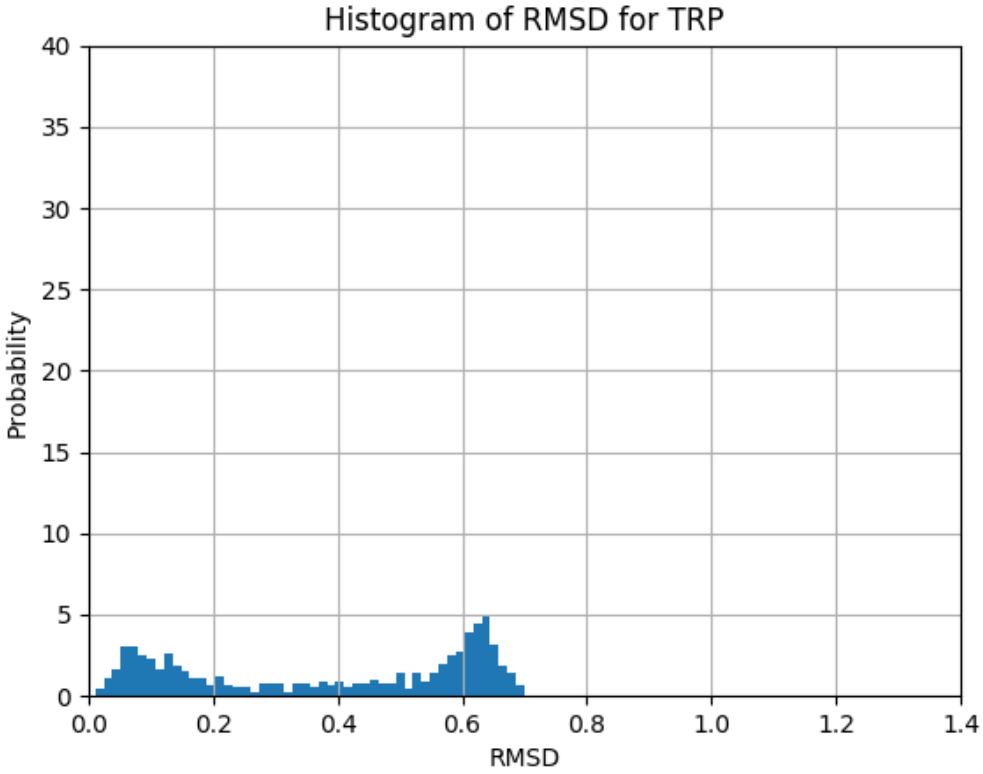

Histogram of RMSD for HIS
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

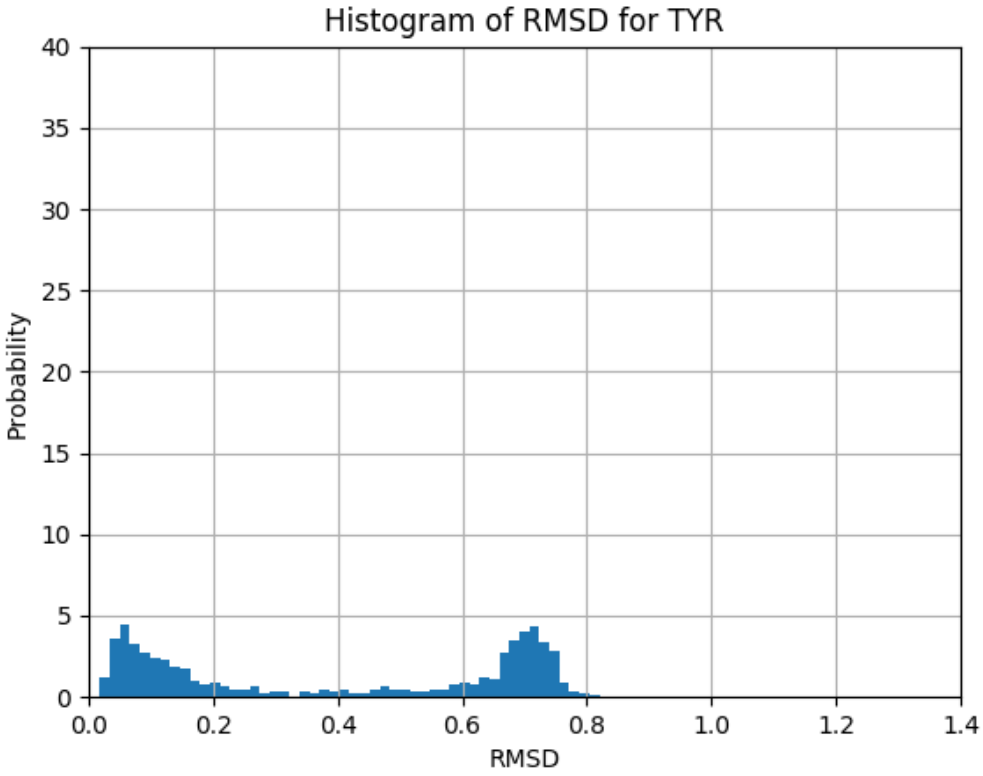

Histogram of RMSD for ILE
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

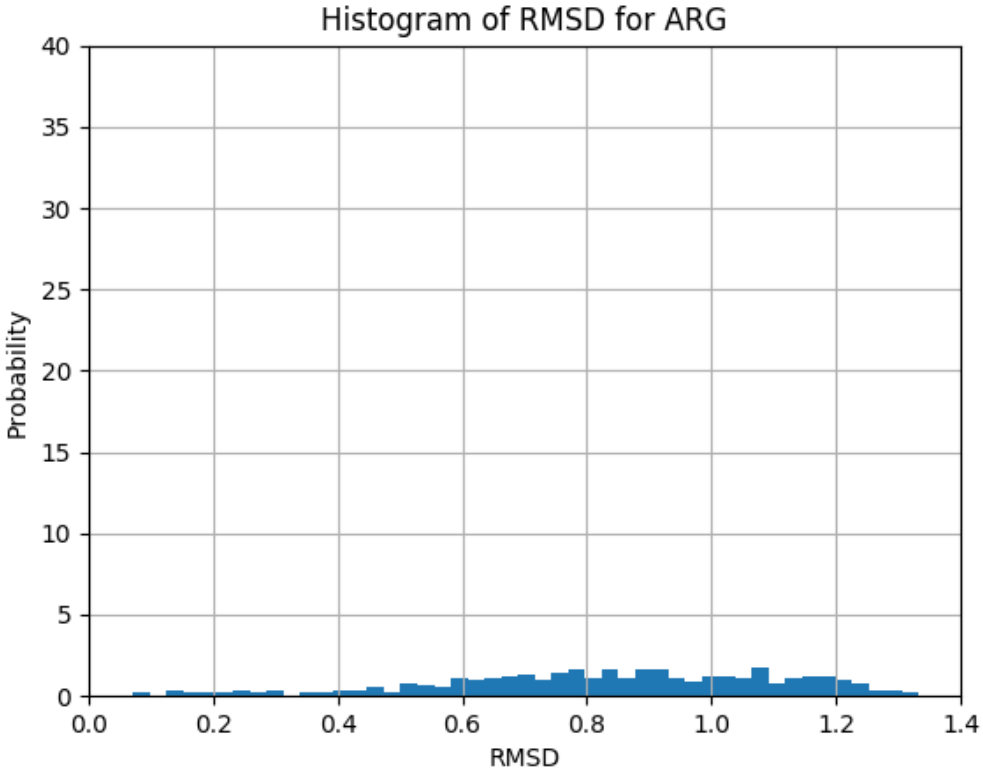

Histogram of RMSD for LEU
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

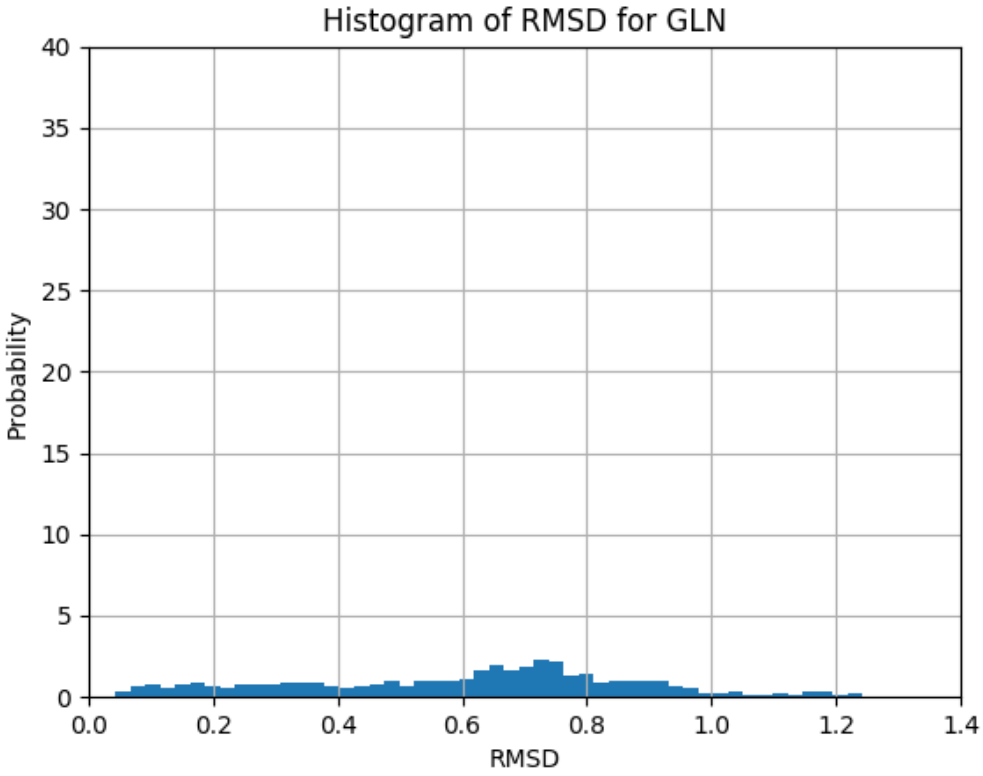

Histogram of RMSD for PHE
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

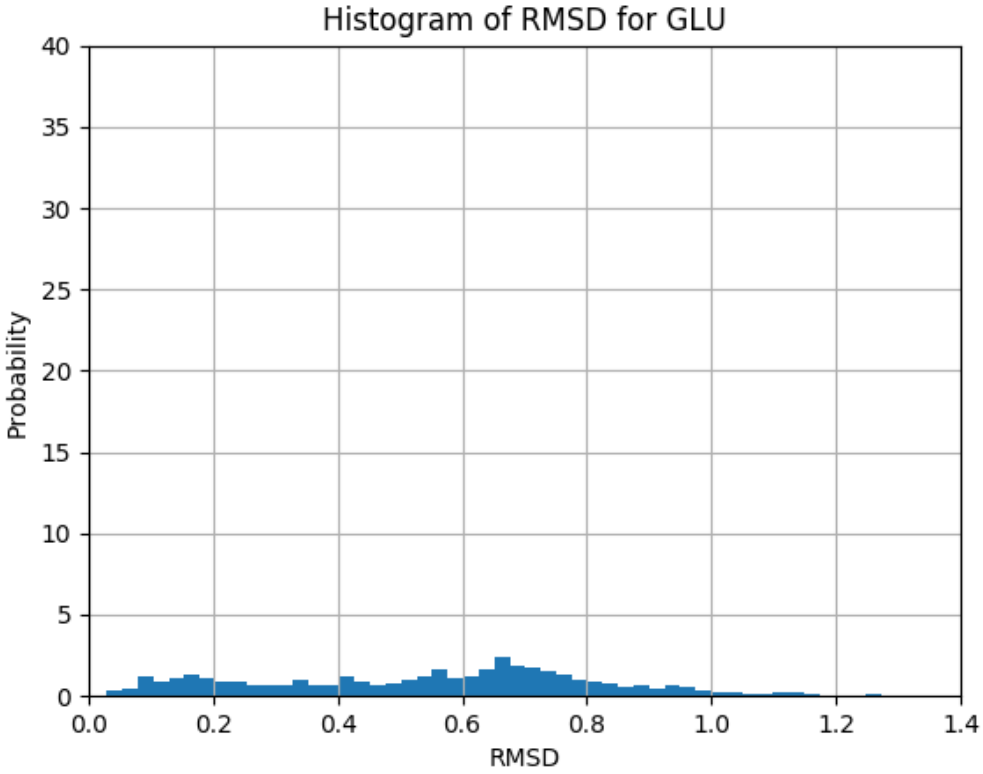

Histogram of RMSD for PRO
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

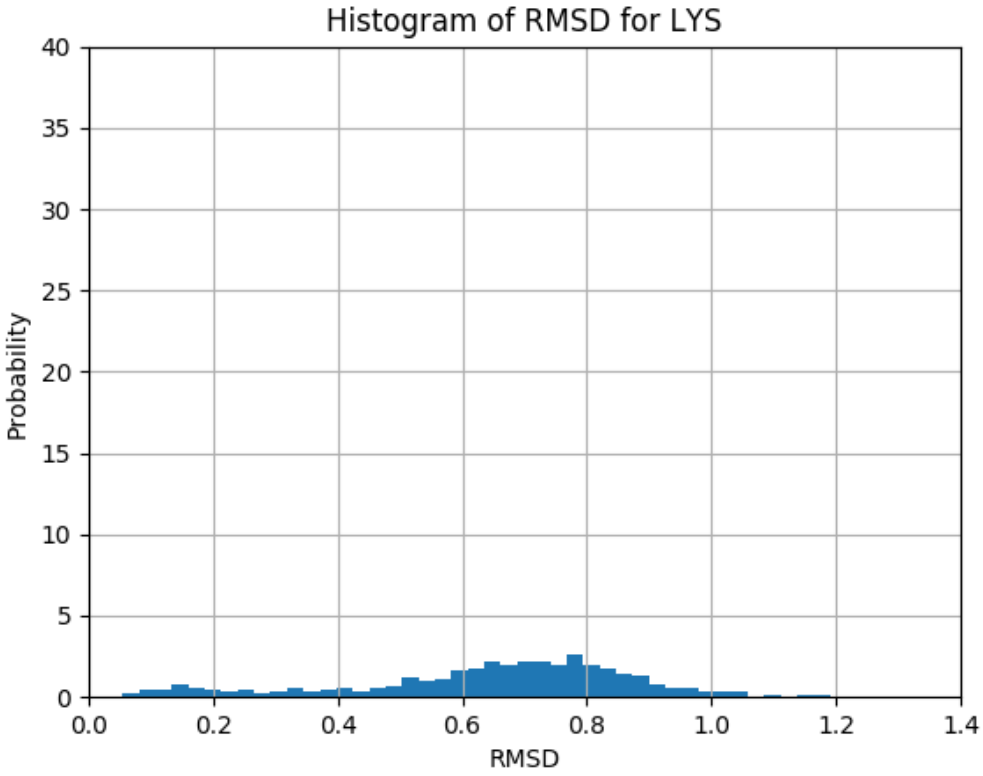

Histogram of RMSD for TRP
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

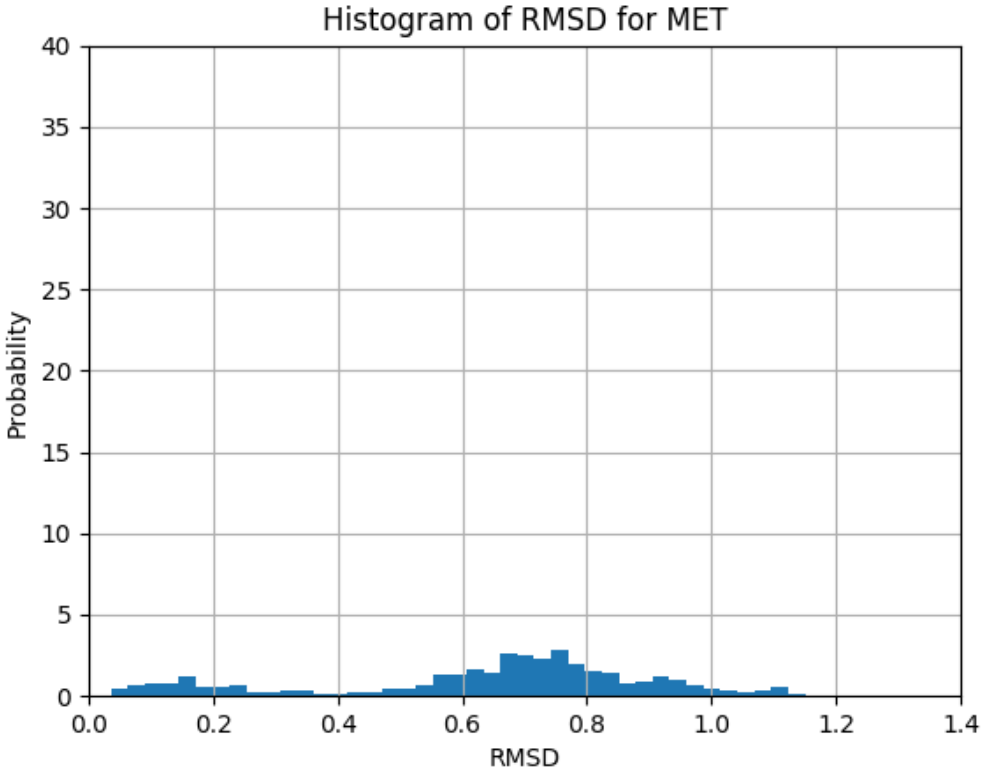

Histogram of RMSD for TYR
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

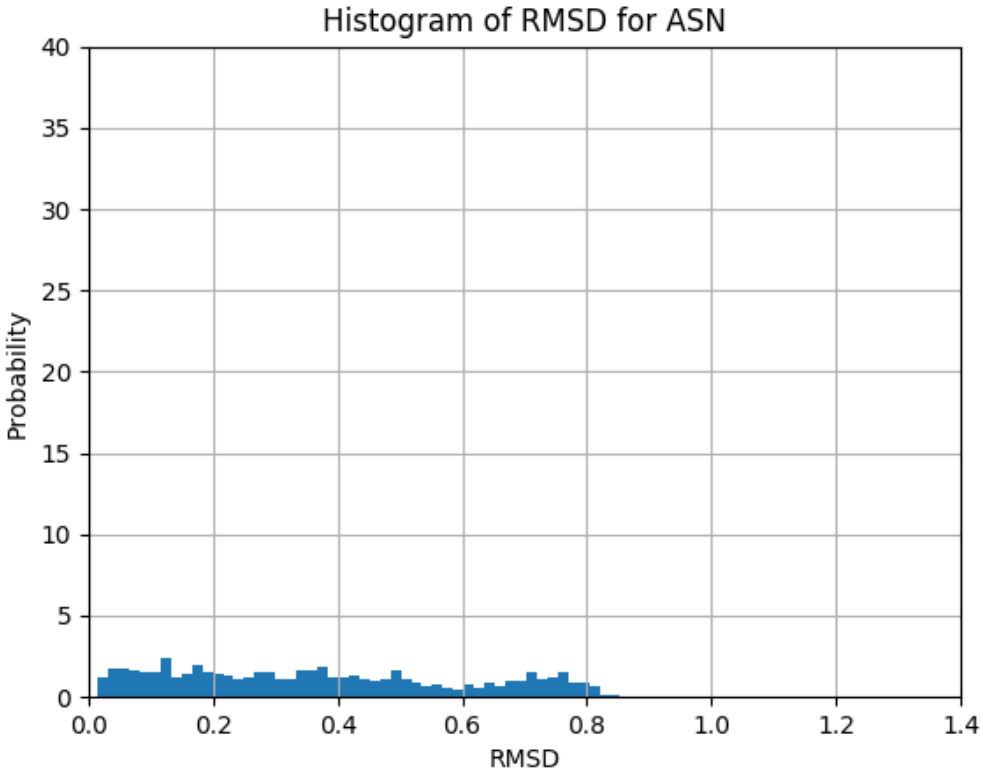

Histogram of RMSD for ARG
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

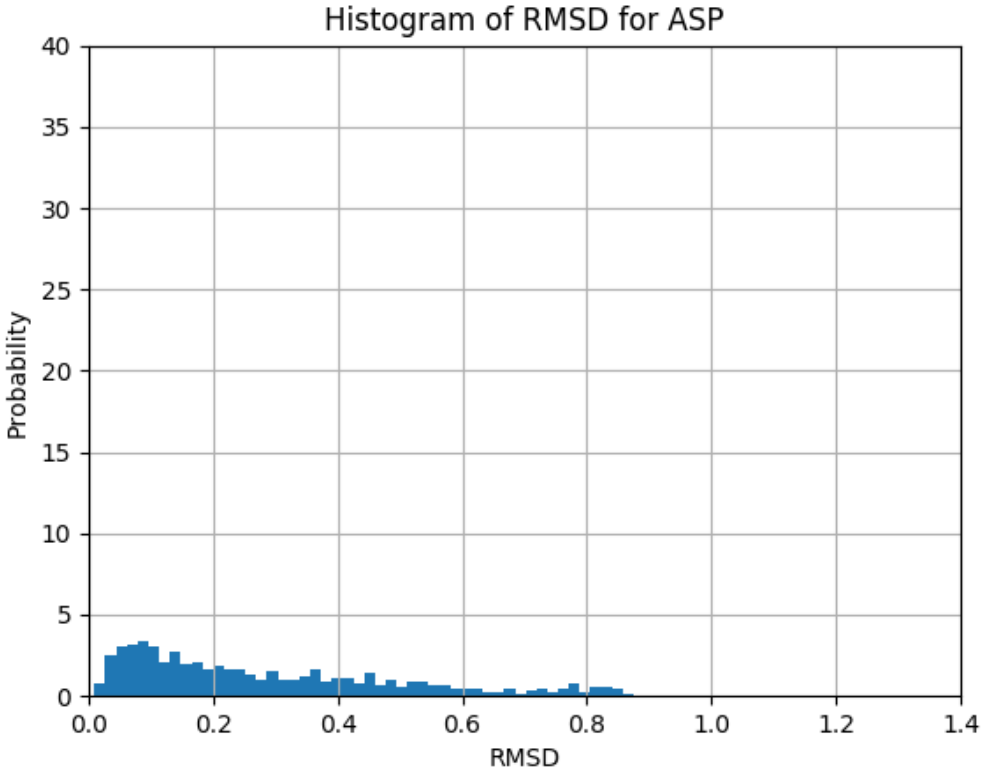

Histogram of RMSD for GLN
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Histogram of RMSD for GLU
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Histogram of RMSD for LYS
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Histogram of RMSD for MET
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

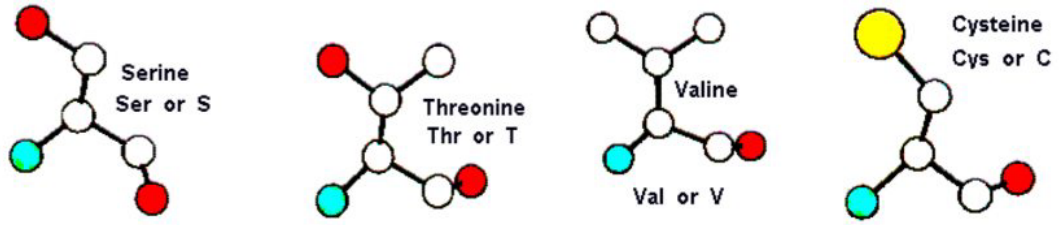

Histogram of RMSD for ASN
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Histogram of RMSD for ASP
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

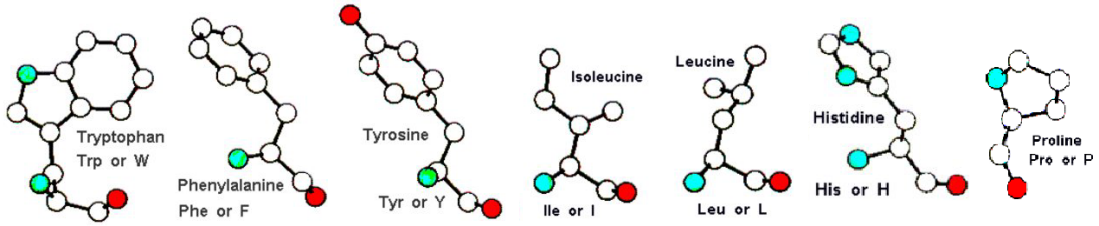
## 4. Conclusions

It is possible to observe that the distributions of RMSD scores exibit certain patterns depending on the constraints present in the structure of the amino acids and other characteristics, like the preference for being shielded from the solvent.

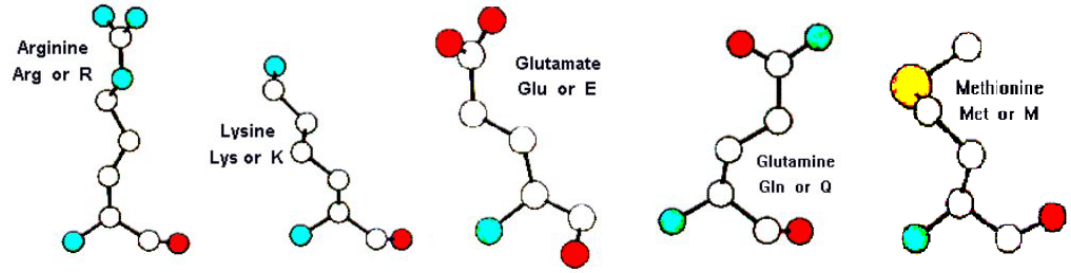
Serine, Threonine, Valine and Cystein have the smallest side chain and they have a more compact structure. As expected they have the lowest RMSD score and exibit the lowest variability in the score, meaning that they have the lowest ability to move in the three-dimensional space.




Tryptophan, Phenylalanine, Tyrosine, Isoleucine, Leucine, Proline and Histidine are the amino acids that present a characteristic pattern in their structural variability. In particular they show a clear bimodal distribution of their RMSD scores, that may indicate that they can be found in two main different structural conformations. With some exceptions the amino acids that show this characteristic pattern are mainly hydrophobic, but this does not seems to be correlated to the bimodal nature of their RMSD distributions. One explanation for the two peaks in the distributions could be that, if a given amino acid is amphipatic, it would be found in two main different structural arrangements depending on the interactions it would have with the surrounding atoms. This explanation anyway doesn't explain the presence of the two peaks in the non amphipatic residues.




Arginine, Lysine, Glutamate, Glutamine and Methionine are the amino acids that present the largest RMSD scores and have the largest variability in their distributions. With the exception of Methionine they are polar amino acids, they all have long side chains and, as it is possible to observe, due to the nature of their structure they have less constraints than others amino acids and they are therefore more flexible and have more freedom of movement.



## References

[1] S.C. Lovell, I.W. Davis, W.B. Arendall III, P.I.W. de Bakker, J.M. Word, M.G. Prisant, J.S. Richardson, and D.C. Richardson (2003) "Structure Validation by $C$ Geometry: , and $C$ Deviation" Proteins: Structure, Function and Genetics 50:437-450.

[2] H.M. Berman, K. Henrick, H. Nakamura (2003) Announcing the worldwide Protein Data Bank Nature Structural Biology 10 (12): 980.

[3] Hamelryck, T., Manderick, B. (2003) PDB parser and structure class implemented in Python. Bioinformatics 19: 2308–2310

[4] Thomas Hamelryck. Associate professor, Computational and RNA Biology; University of Copenhagen.