Protein practical part

January 21, 2020

1 The protein practical part

1.1 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.

1.2 Materials and methods

For the exercise we used the Top500 database of PDB files, available from http://kinemage.biochem.duke.edu. 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].

1.2.1 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: 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).

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

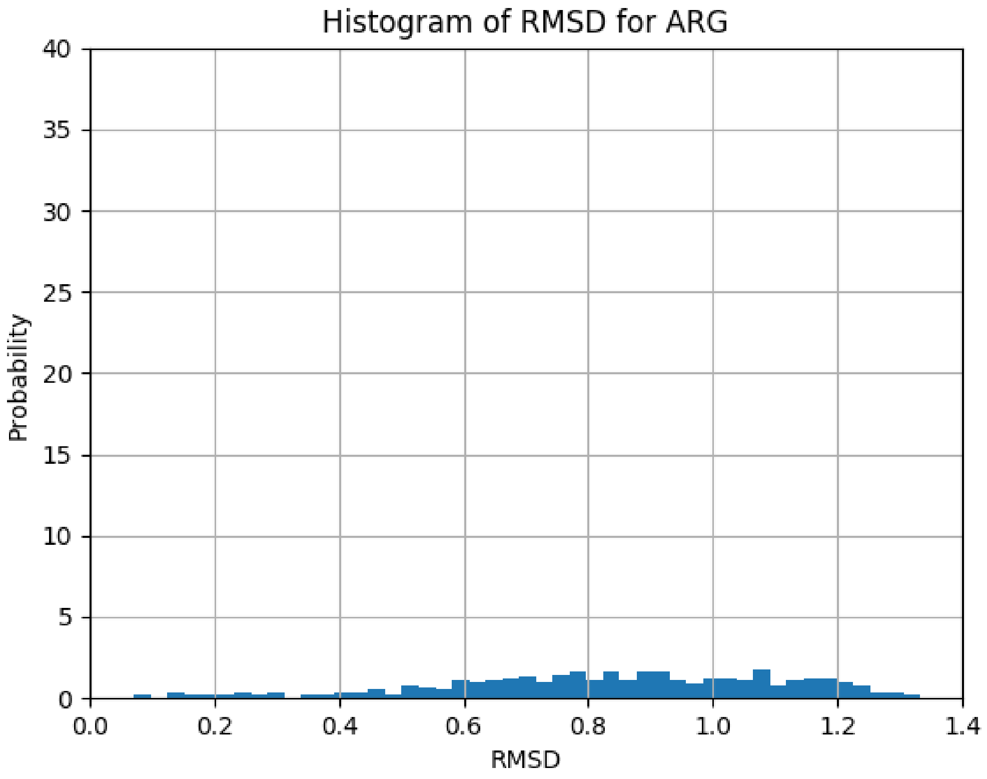
Finally I centered the atoms by subtracting the center of mass to each coordinate vector in the set. The implementation of the RMSD algorithm (and most of the rest of the code) is from our structural bioinfomatics professor Thomas Hamelryck. In order to find the rotation matrix U, I applyed the singular value decomposition (SVD) to the correlation matrix R.

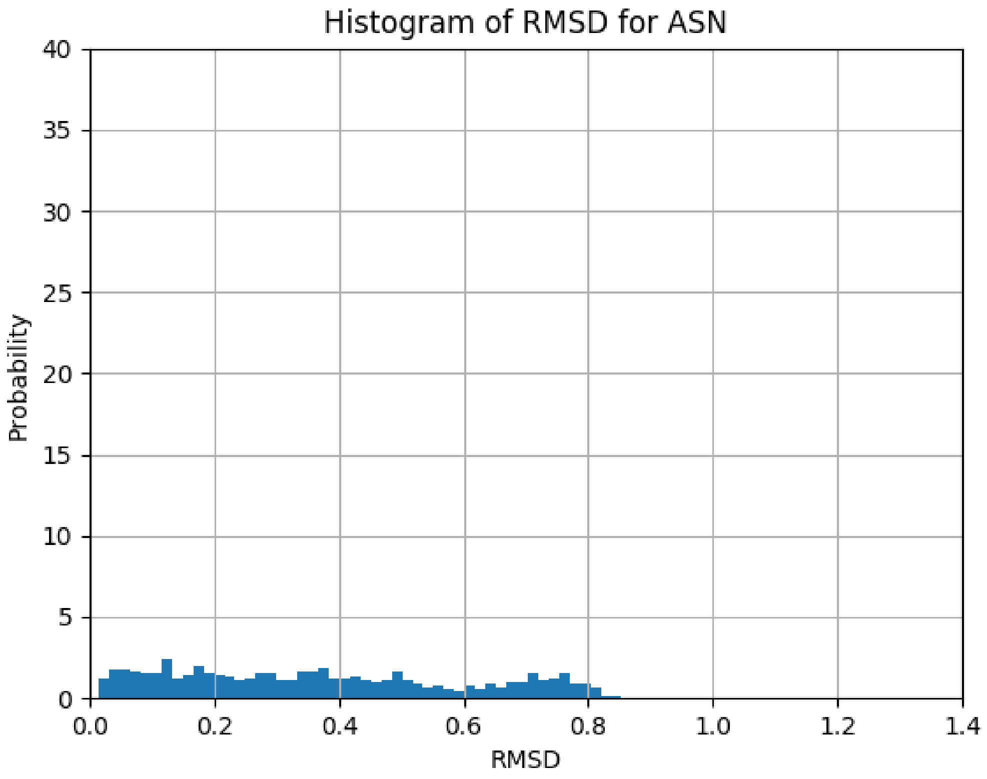
$$R = YX^t = VSW^t \tag{3}$$

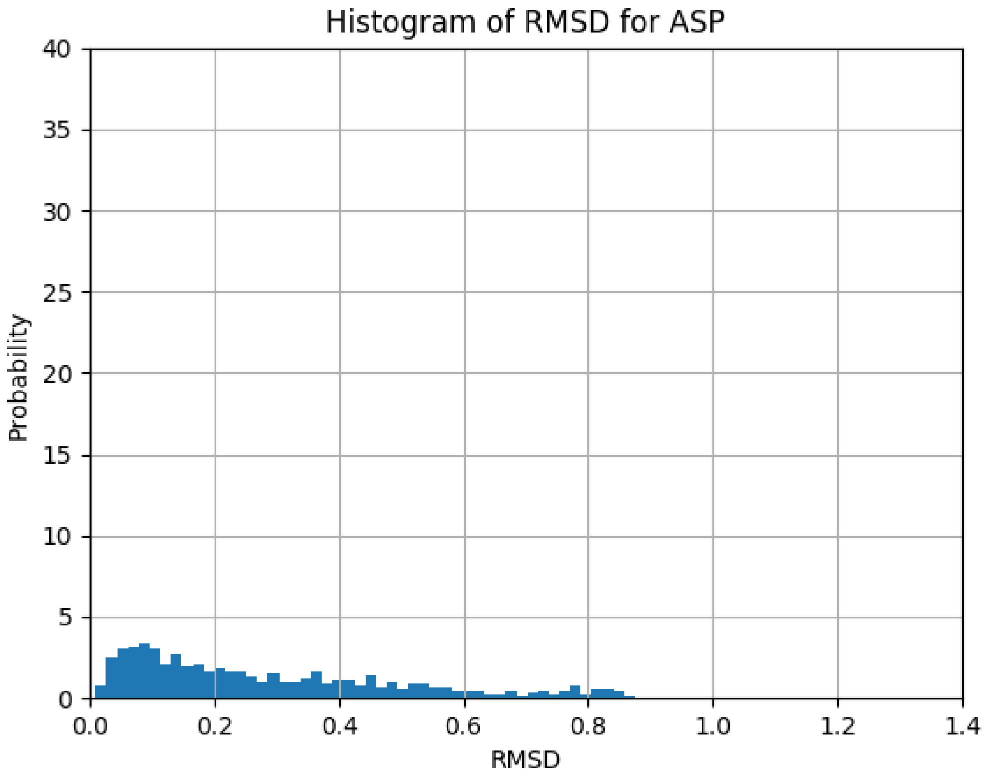
$$U_{\min} = WV^t \tag{4}$$

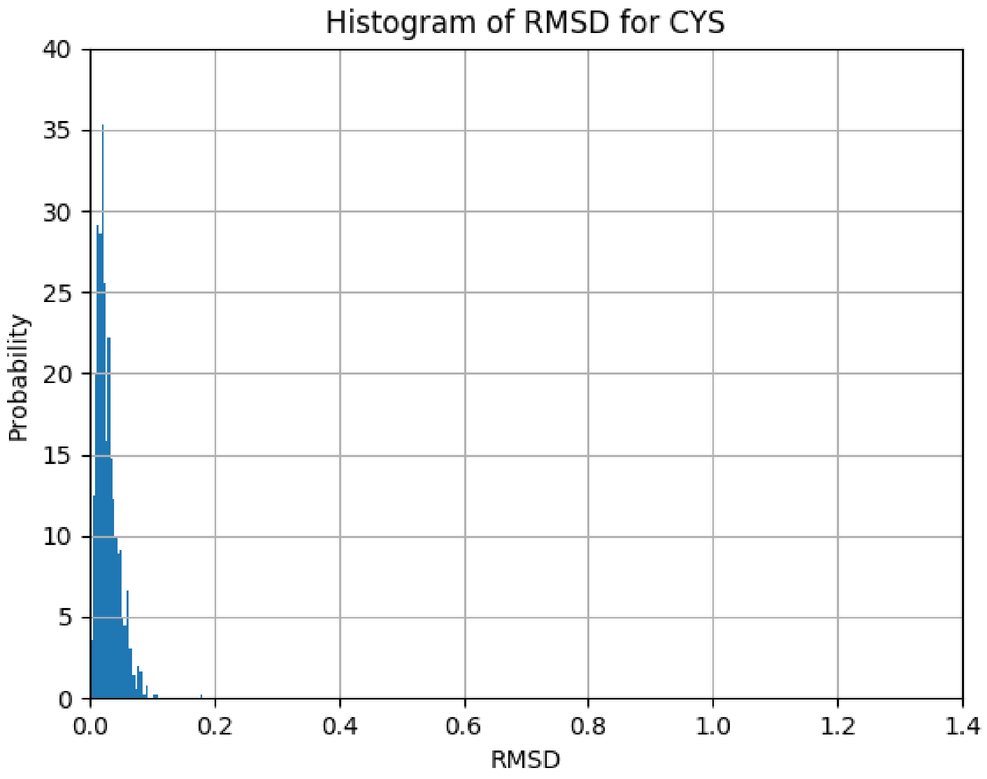
Sometimes the rotation matrix U that minimize the RMSD is a roto-inversion, that will superimpose a mirror image. To avoid that we have to multiply the components of the rotation matrix U for Z = diag(1,1,-1), and we also change the sign to the third element of the diagonal matrix S. Than I applyed the rotation matrix U to y and I finally calculated the RMSD from the set of coordinates.

1.3 Results

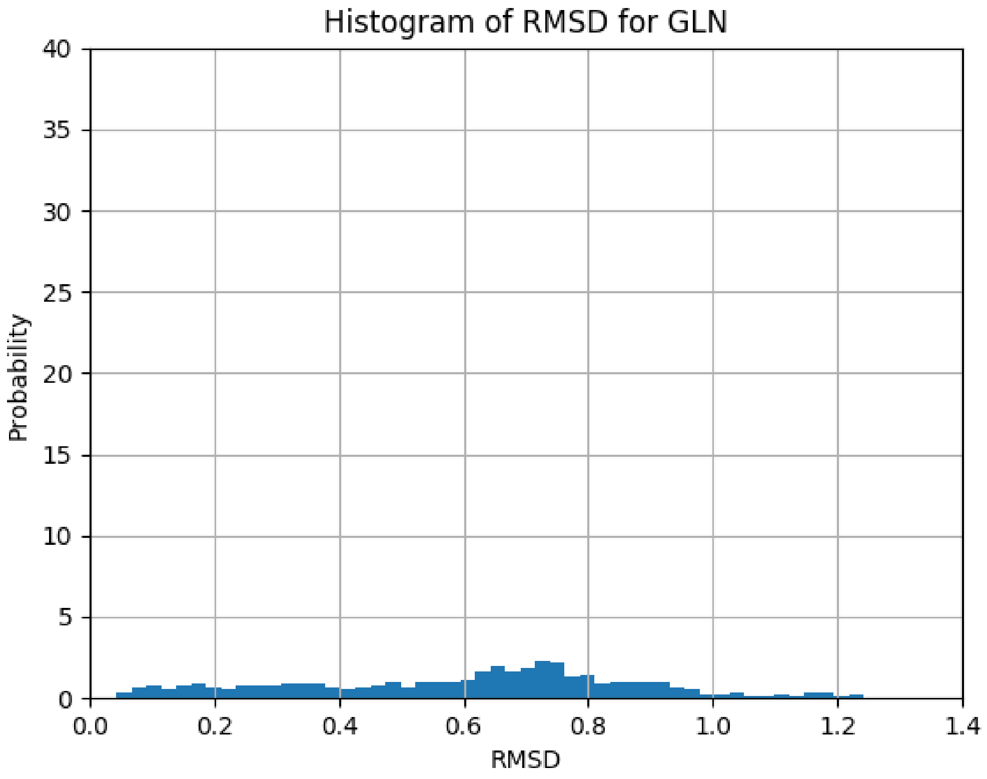

Histogram of RMSD for ARG
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Histogram of RMSD for 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

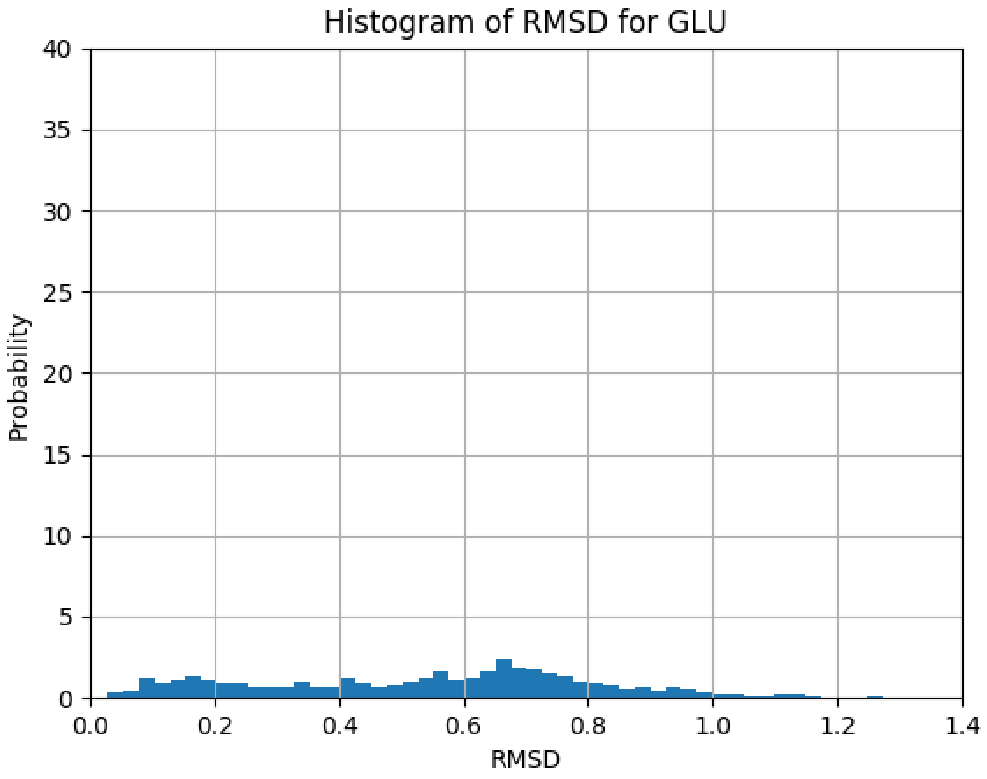

Histogram of RMSD for ASP
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

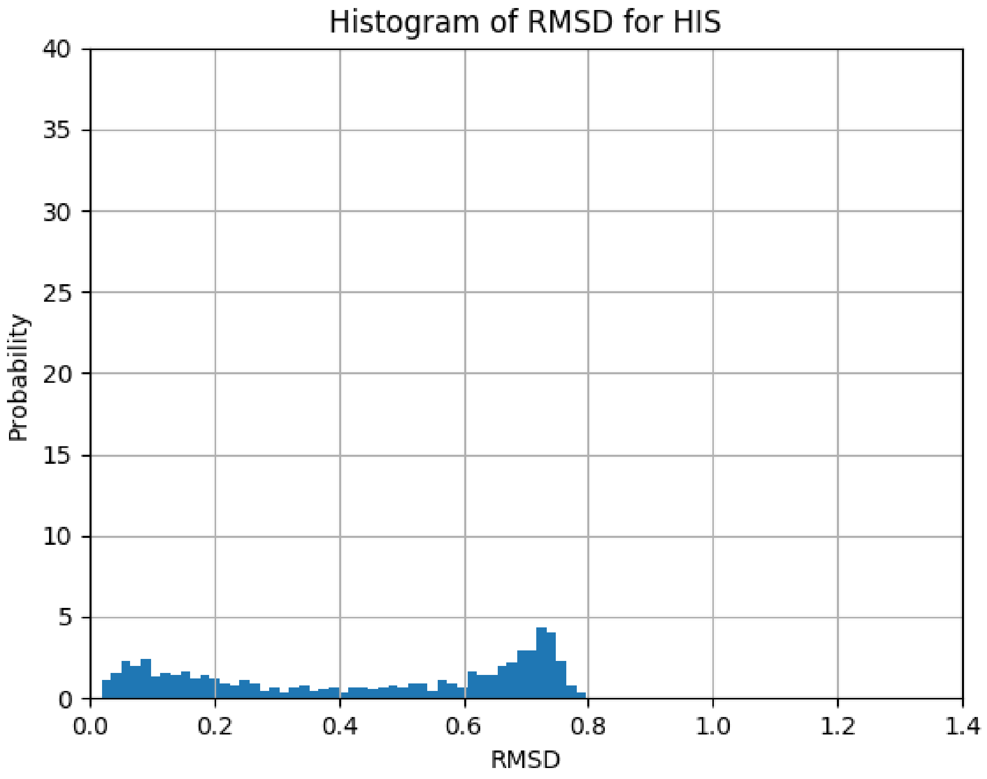


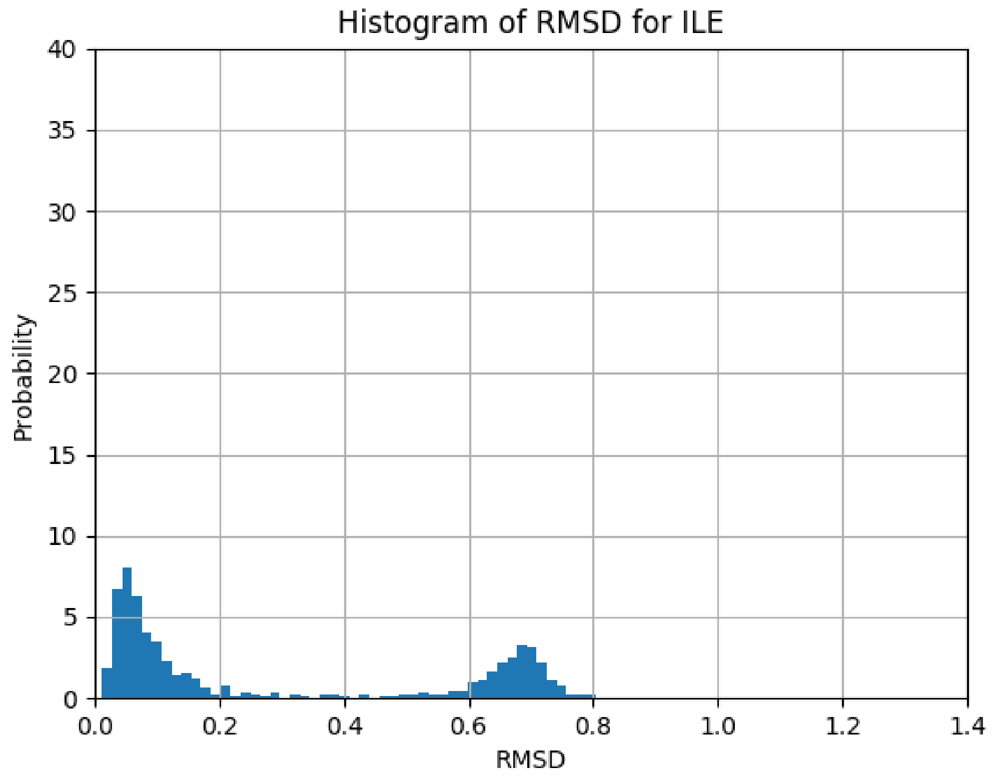
Histogram of RMSD for CYS
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

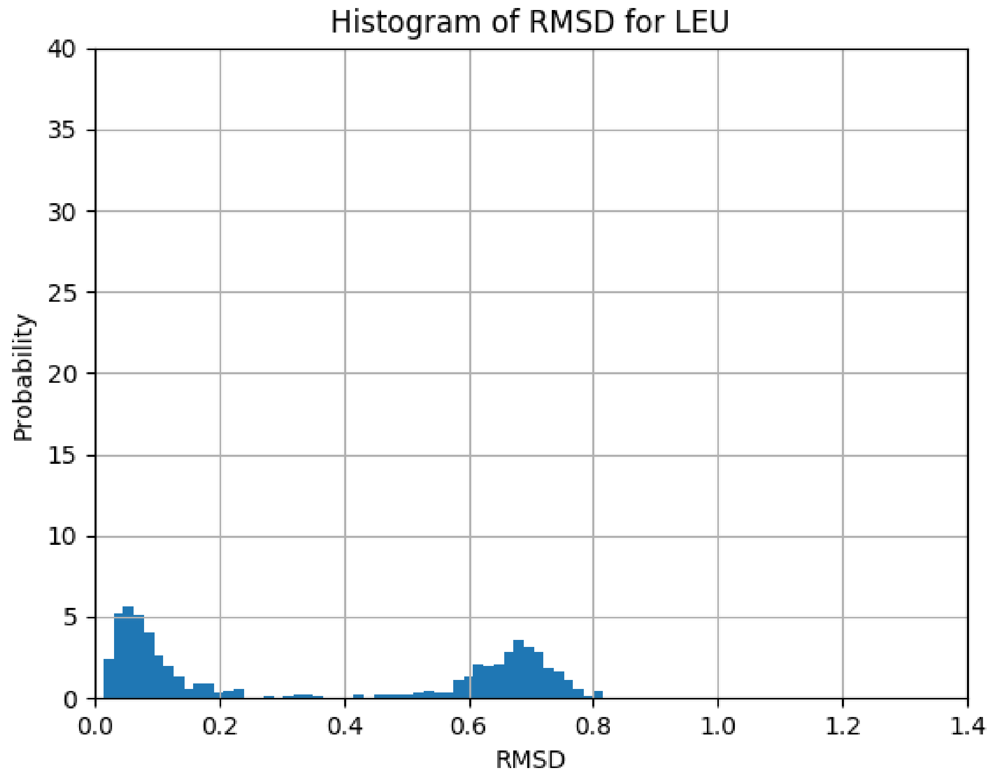


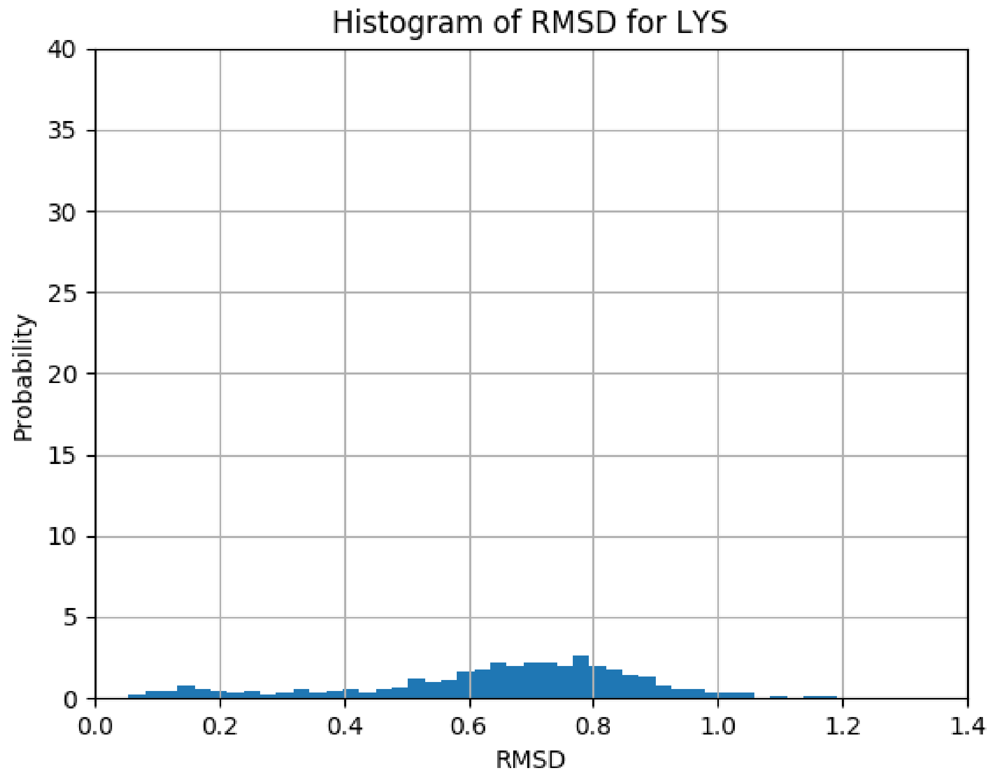
Histogram of RMSD for GLN
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

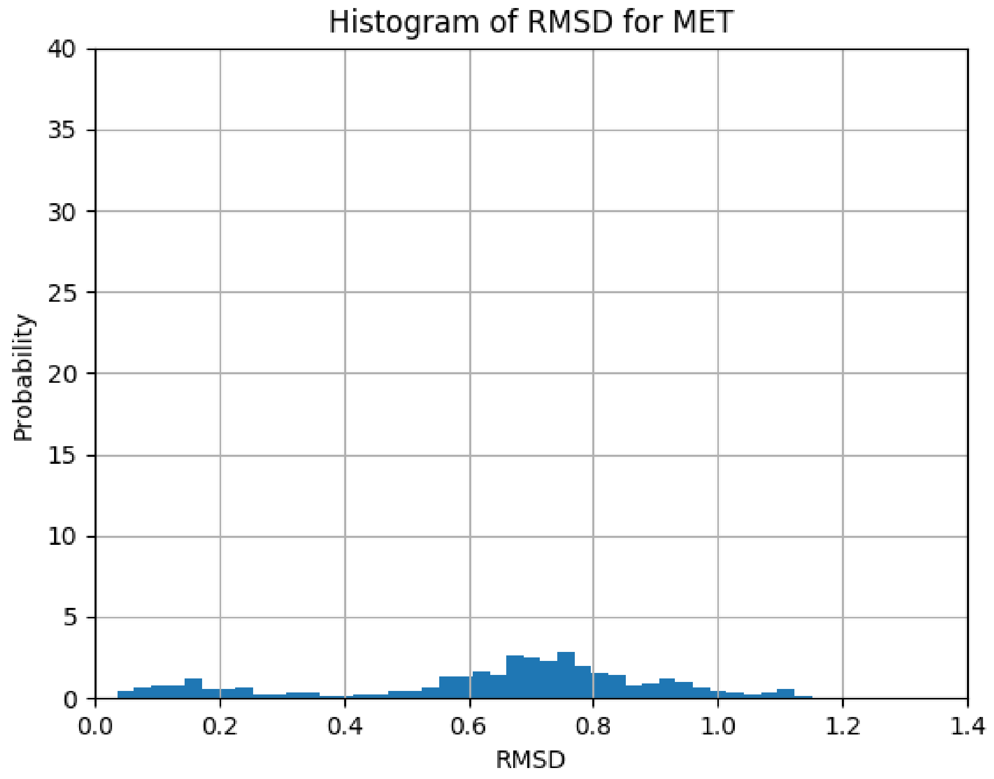

Histogram of RMSD for GLU
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

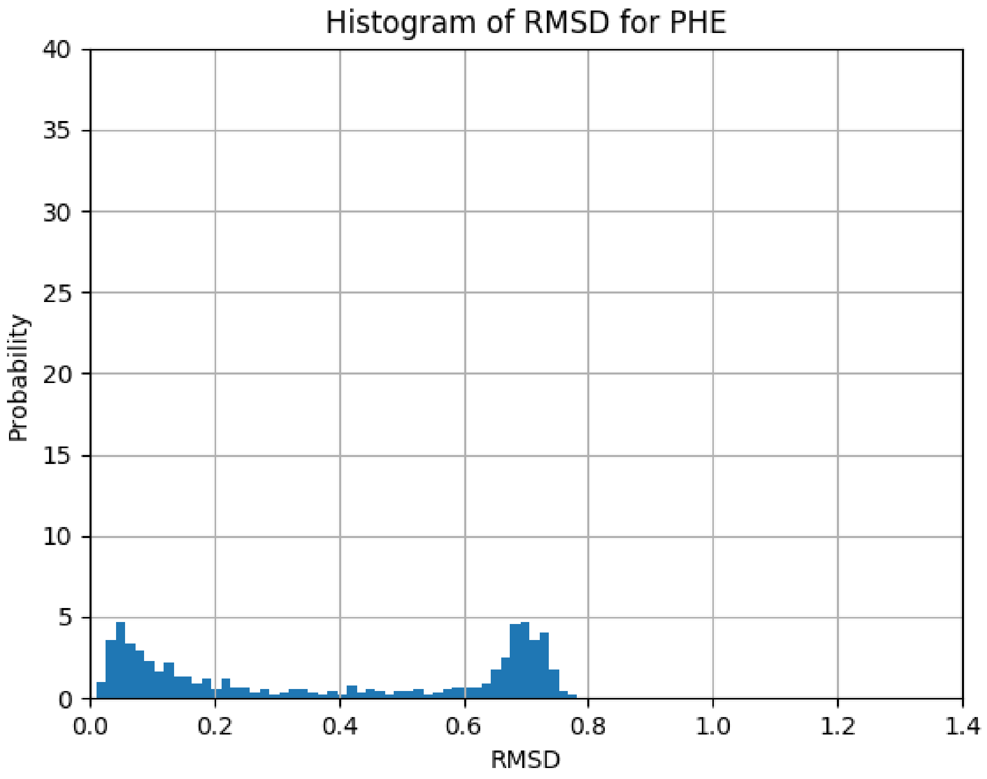

Histogram of RMSD for HIS
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

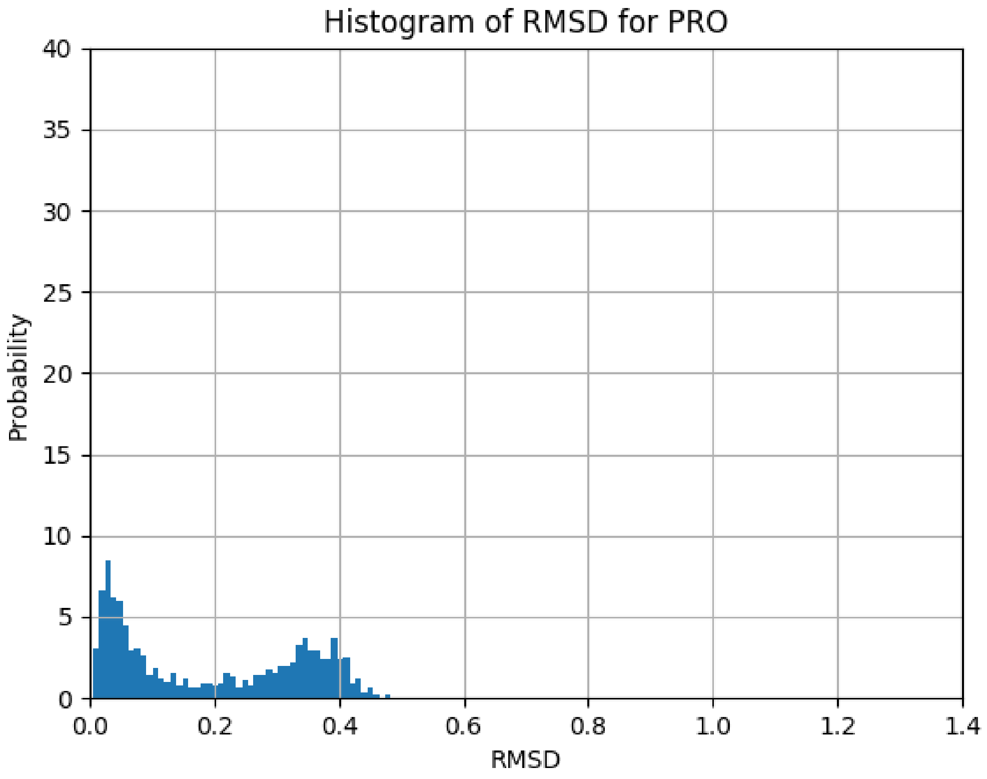

Histogram of RMSD for ILE
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

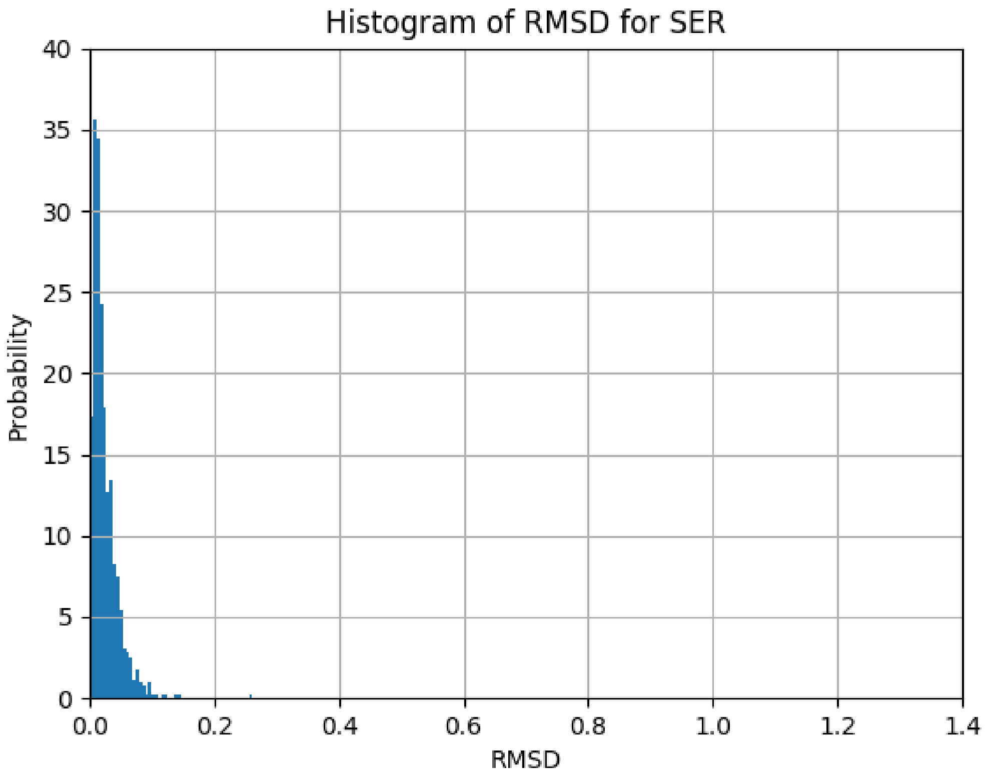

Histogram of RMSD for LEU
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

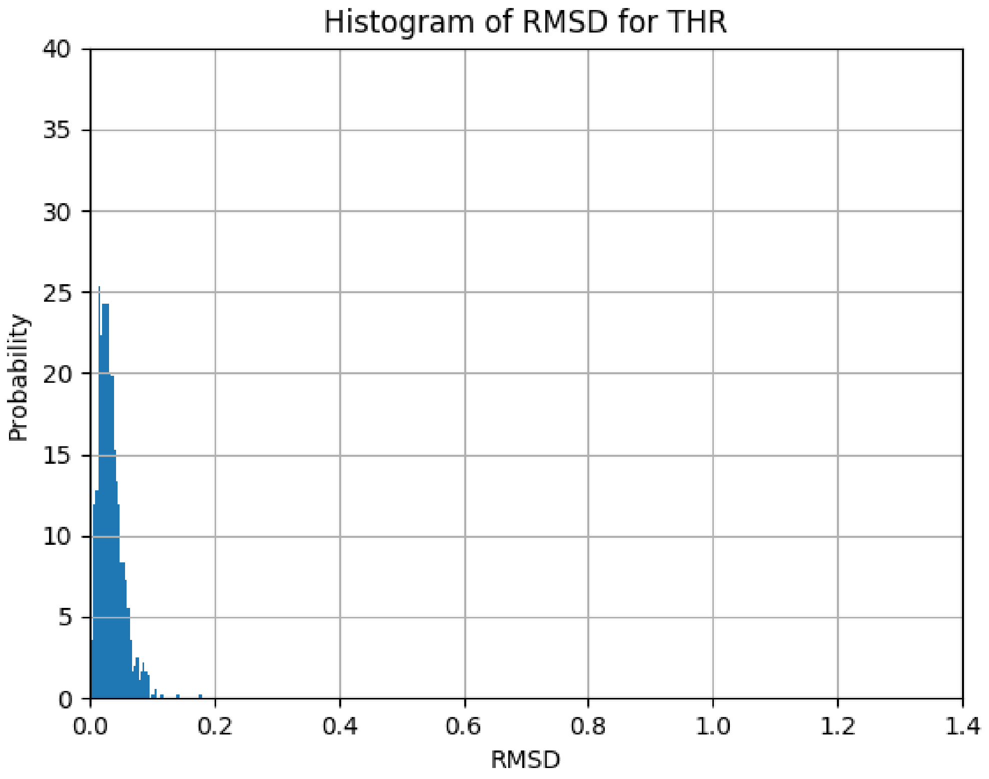

Histogram of RMSD for LYS
Probability
RMSD
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
5
10
15
20
25
30
35
40

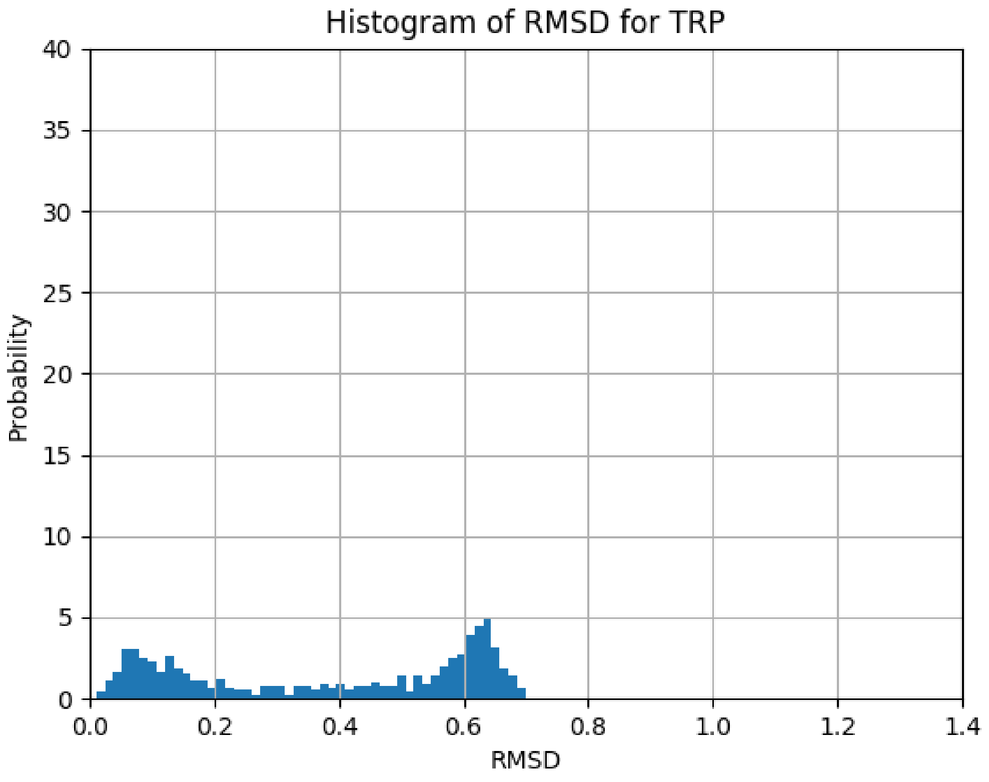

Histogram of RMSD for MET
Probability
RMSD
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
5
10
15
20
25
30
35
40

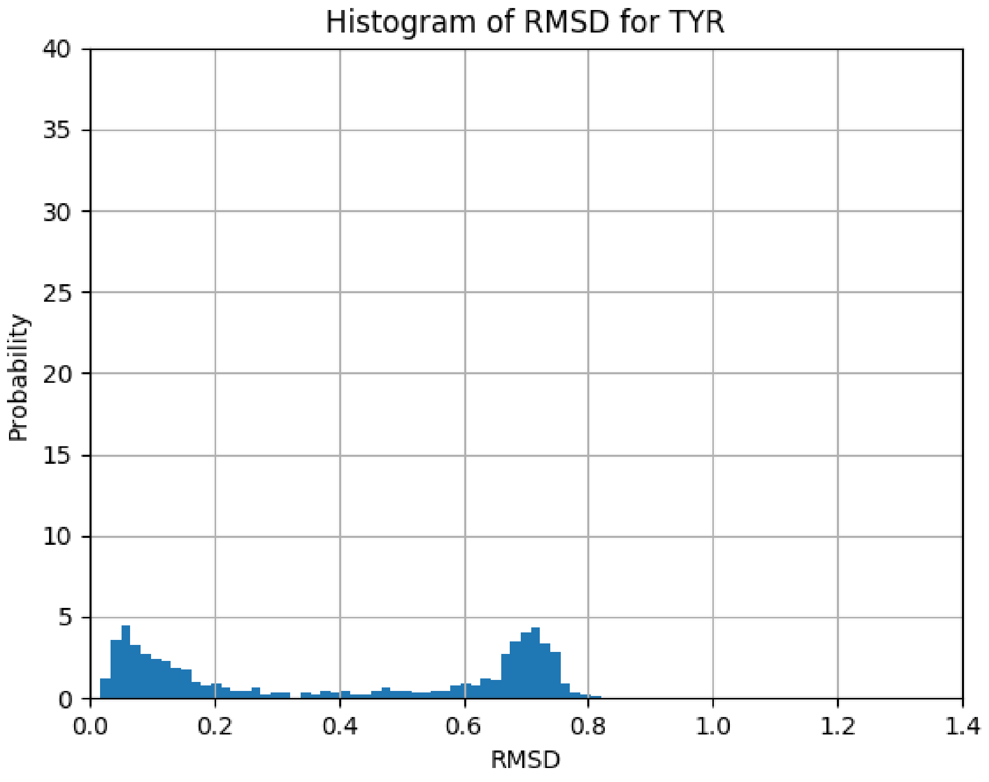

Histogram of RMSD for PHE
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

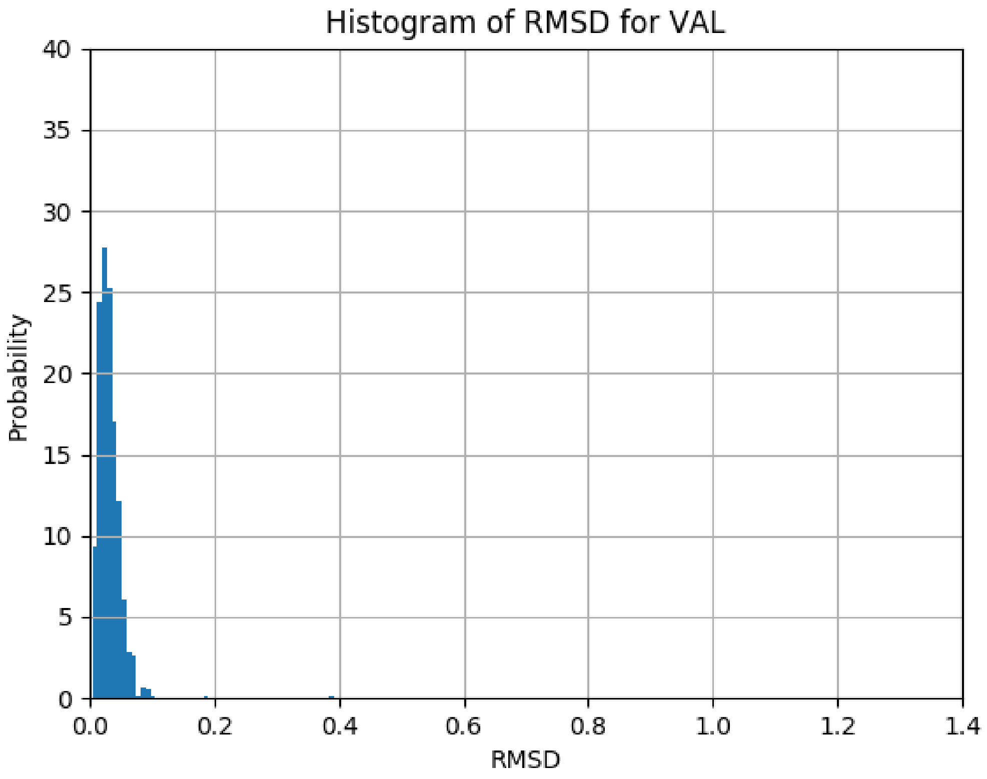

Histogram of RMSD for PRO
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4


Histogram of RMSD for 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 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

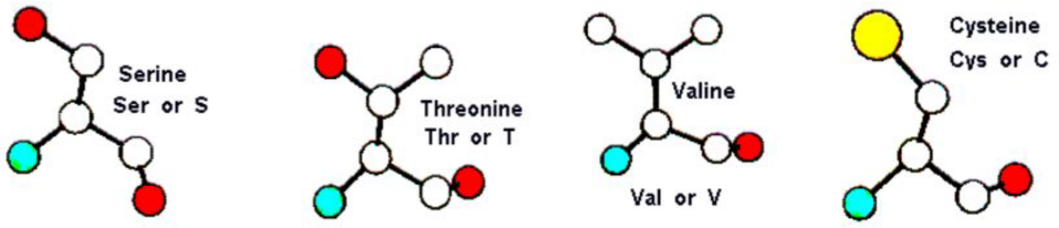

Histogram of RMSD for TYR
Probability
RMSD
0
5
10
15
20
25
30
35
40
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4



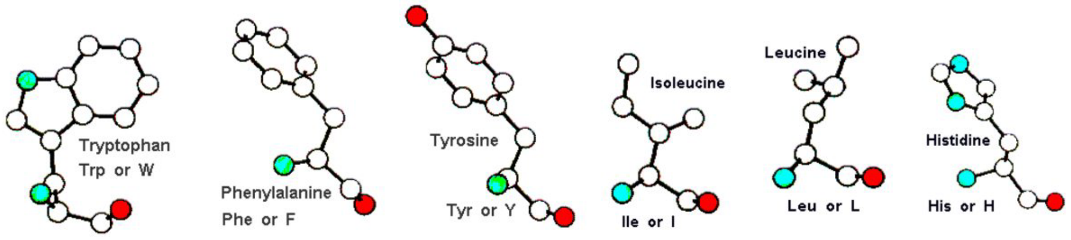
1.4 Conclusions

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.

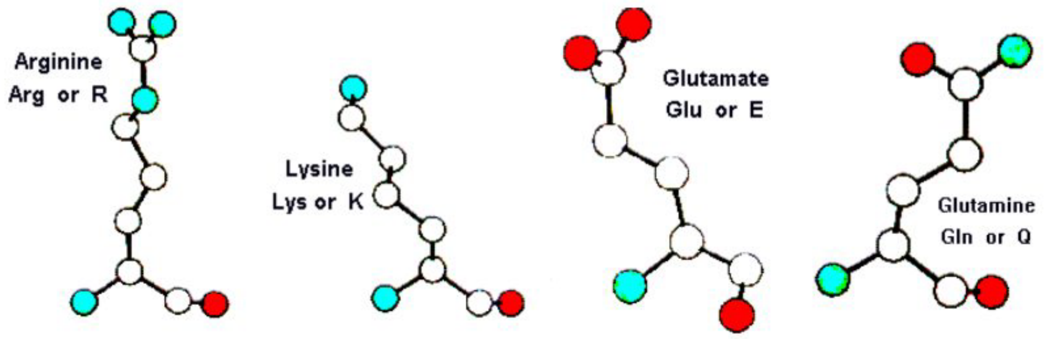
Serine, Threonine, Valine and Cystein are the smallest amino acids with a more compact structure. As expected they have the lowest RMSD score and exibit the lowest variability in the score.



Tryptophan, Phenylalanine, Tyrosine, Isoleucine, Leucine and Histidine are the amino acids with more extreme structural variability. In particular they show a clear bimodal distribution of their RMSD score, that may indicate that they can be found in two main different structural conformation. The amino acids that show this characteristic pattern are mainly hydrophobic, and the large variability may be due to the fact that they are located in proteins core. In that case the spatial arrangement of their atoms needs to change in relation to the different environmental constraints found in different proteins. But this does not explain the bimodal nature of their RMSD distributions, one explanation could be that they have amphipatic intercations. This explanation fit the case of Tryptophan and Tyrosine which are amphipatic and Histidine which may be ambivalent and often buried in an unprotonated form.



Arginine, Lysine, Glutamate, Glutamine are the amino acids that present the largest RMSD score with a similar Gaussian distribution. They are polar amino acids and as it is possible to observe, due to the nature of their structures, that they have less constraints than the others amino acids and are therefore more flexible.



1.5 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