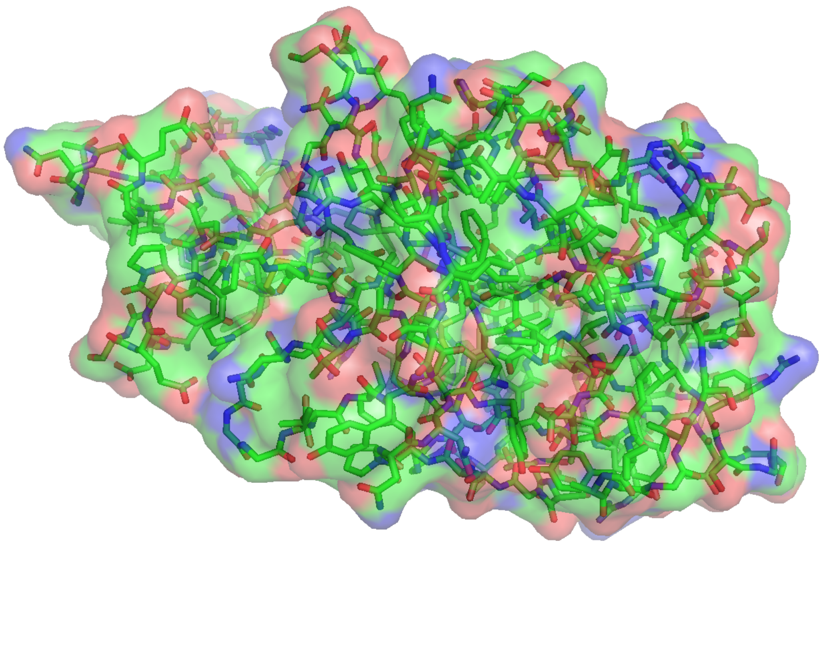
Structural Bioinformatics Final Exam
Protein Part

ANTONIO ORTEGA JIMÉNEZ
University of Copenhagen
January 27, 2017

RMSD analysis on identical n-mers in the top100H database



Contents

1 Introduction

The study of protein structure is a hot topic on several fields like medicine and the biotech industry. For that matter, Bioinformatics methods that predict and analyze structure information are constantly being developed by the scientific community. Moreover, since research groups all around the world publish their results on open databases this methods can be easily be applied to real data. One of such databases is the Protein Data Bank (PDB), maintained by an international consortium. In other words, bioinformaticians have both open data and free software to try out new ideas.

Do we expect structure similarity in identical sequences of n residues? Aminoacidic sequence has a great impact on secondary structure, up to making some conformations highly unlikely. This is due to the particular spatial properties and constraints each different kind of aminoacid displays. Thus, we should expect a general trend toward similarity in random pairs of short lengths, for short sequences acquire very little degree of 'conformational'freedom. It follows that, as the length of the fragments grows, only sequence matches should remain similar.

On the other hand, structural similarity between a pair of structures can be modelled using the Root Mean Square Deviation (RMSD) statistic. While a small value of this parameter tells us that the pair of structures has high similarity, a big value indicates no similarity at all.

In this report, a simple *in silico* analysis of protein structure data was run to try to give an answer to this question. The Biopython module, contained in the Python programming language, together with the statistical language R, were deployed in the extraction of all possible n-mers in a protein database and in the following computation of their RMSD values.

2 Materials and methods

2.1 Structure and sequence parse, and generation of n-mers

Classroom data are like teddy bears and real data are like a grizzley bear with salmon blood dripping out its mouth.

Jenny Bryan

Real data come messy and in ways that make very hard to learn knowledge from it. The dataset provided in this report fitted one of those grizzley bears, for the reason explained in *Rebuild of the top100H*.

Structural (spatial) information was parsed from the database by making use of the `PDBParser.get_structure()` method, whereas sequence data was parsed using the `SeqIO.parse()` method. Both of these functions are available in the Biopython package [1]. Chain breaks, consisting of residues whose spatial information is either missing, or irrelevant (due to their disordered nature), were handled with several user defined functions. In a nutshell, this chain breaks were represented in the sequence set by replacing the letter of the affected aminoacids with X.

Fragments were generated with Python list's comprehension:

```
[(seq[i:i + n], i) for i in range(0, len(seq) - n) \
if not "X" in seq[i:i + n]]
```

This way, subsequences of length *n* that do not contain X are stored in a list. This list represents all n-mers that can be extracted from sequence *seq*.

Information relating to the n-mer sequence, position and geometrical coordinates was stored in a customized data type called `Fragment`:

```
class Fragment(object):

    def __init__(self, seq, start, seq_id, \
    chain_id, X, atoms):
        self.seq = seq              # aminoacidic seq
        self.start = start          # start pos
        self.seq_id = seq_id        # pdb id of the protein
        self.chain_id = chain_id    # chain id
        self.X = X                  # coordinates matrix
        self.atoms = atoms
```

2.2 Rebuild of the top100H

The top100H database consists of a collection of a 100 files that represent a 100 different protein structures [2]. It is available in this link:

http://kinemage.biochem.duke.edu/databases/top100.php

A first usage of these files revealed several issues related to the out of date syntax that was employed in their creation. For that reason, the database was rebuilt using modern day files. These new pdb files were downloaded from the PDB [3] using the `pdbList.retrieve_pdb_file()` method, available in Biopython. Entry ids were parsed from the ones in the original database. The scripts used to complete this are available at

`https://github.com/antortjim/rmsd_nmers/tree/master/database`.

2.3 Pairwise computation of RMSD

The computation of RMSD [4] receives two matrices of dimensions described as below:

- Each column stores a coordinate vector that represents the position of a spatial entity. In this report, only Cα atoms were used.
- Each row stores one of the 3 dimensions in space.

Thus, in this report, n x 3 matrices were used, where every row represents the $C\alpha$ atom in each residue of the n-mer.

A structurally relevant RMSD has to be computed on two matrices X and Y that have been centered and rotated so that the RMSD is minimized. This requirement is fulfilled by means of a mathematical method called Singular Value Decomposition (SVD). This algorithm computes the rotational matrix required by Y to minimise RMSD. Together with a centering method, the RMSD can be brought to its minimum.

These are the key aspects of this combined SVD + RMSD calculation algorithm:

- Matrices are centered in the coordinates origin by substracting the center matrix (1 x 3) to all rows in the coordinates matrices. The center of the new matrices will now be 0, 0, 0.
- Compute the correlation matrix R between X and Y (XT^t).
- Perform SVD on this matrix and return 3 matrices V, S, and W^t.
- The minimum RMSD can now be computed using the following formula.

$$RMSD = \sqrt{\frac{1}{n}(E_0 - 2(\sigma_1 + \sigma_2 + \sigma_3)}$$

where

$$E_0 = \sum_{i=0}^{n-1}(|x_i|^2 + |y_i|^2)$$

and the σ summation is the sum of the S matrix.

This whole computation was achieved by making use of user defined functions and the svd implementation available at the Python module numpy [5] (`numpy.linalg.svd()`).

User defined functions may be found in this link

`https://github.com/antortjim/rmsd_nmers/blob/master/user_defined.py`

2.4 Visualization of data

Interpretation and visualization of the data were performed in R using the ggplot2, data.table, dplyr, cowplot and Shiny [6] packages. Briefly, the data generated with the Python scripts was loaded with R's `read.csv()`. The random and sequence pairs data were merged using `dplyr::full_join()`. Duplicates were detected using data.table methods and finally, graphics were generated using ggplot2 [7]. The interactive website was built with Shiny.

3 Results

3.1 RMSD values distribution for 5-mers

As previously explained, the RMSD values obtained in real sequence pairs was expected to be significantly different from those obtained in random matches (with no sequence similarity). As shown in figure 1, this was the case.

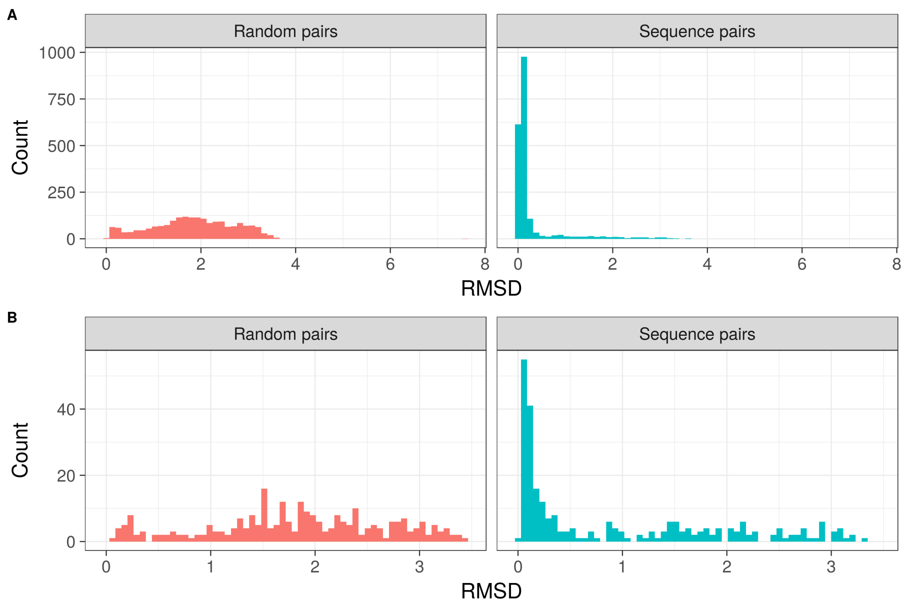


Figure 1: **5-mers RSMD distribution histograms**. **A** If all intraprotein matches and those occuring in structure-sharing proteins are included, an exaggerated count is obtained for low RMSD values. **B** If matches from the same protein are discarded, and only the first of those in structure-sharing proteins are taken into account, a more sensible distribution is obtanined. In both cases, a clear trend toward low values is observed in sequence pairs, whereas no particular trend is observed in sequences paired at random. No peak, besides the one at very low RMSD values, was observed.

A total of 266 fragments fell in the latter category. The median RMSD value for the random pairs was 1.86, and only 0.33 in the sequence pairs.

3.2 Proteins under PDB entries 4ptp and 1mct share a domain

Notably, a huge amount of sequence pairs were found between the same pair of proteins. It was specially true for the pair of proteins with ids 4ptp and 1mct. This pair exhibited 81 5-mer matches.

The existence of this and other matching-prone pairs was easily spotted in the preliminar histogram of the RMSD values (fig 1 A), which showed a extremely high count at very low RMSD values. This is due to the fact that matching-prone pairs most likely share domains, and consequently return a high count of 5-mers with low RMSD values.

The 4ptp and 1mct pair was further explored. First, in order to check the presence of a shared domain, the Pfam database **??** was queried. Indeed, both proteins share a Trypsin domain. Secondly, this domain was visualized using the Pymol suite (fig. 3.2). The alignment of both structures confirms that they share a domain.

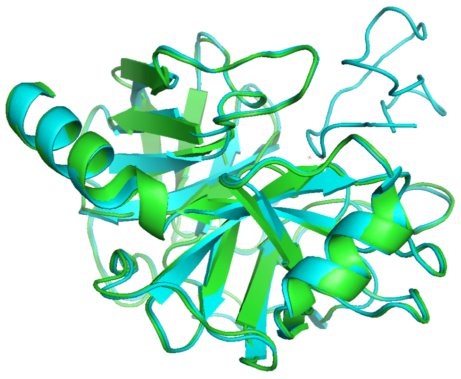


Figure 2: 4ptp and 1mct are highly similar.

3.3 Sequence pairs with high RMSD values adopt opposite conformations

As well as sequence pairs with low RMSD values, many with higher RMSD values were detected. Let us visualize, the pair with sequence ALTAA. This pair is found in two proteins:

- Chain A and residue 64 in protein 1aru.
- Chain A and residue 231 in protein 1nif

This pair exhibited a RMSD value of 3.13, one of the highest for a sequence pair. A visualization of their structure (fig 3.3) reveals the reason behind it: In 1aru, this sequence adopts an α fold, whereas in 1nif, it adopts β conformation.

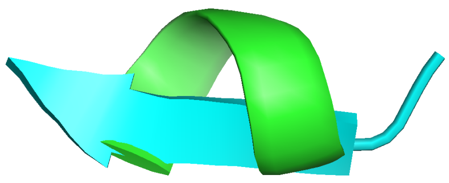


Figure 3: Even though they share aminoacidic sequence, this pair of 5-mers adopts a totally different fold that leads to an RMSD value of 3.13.

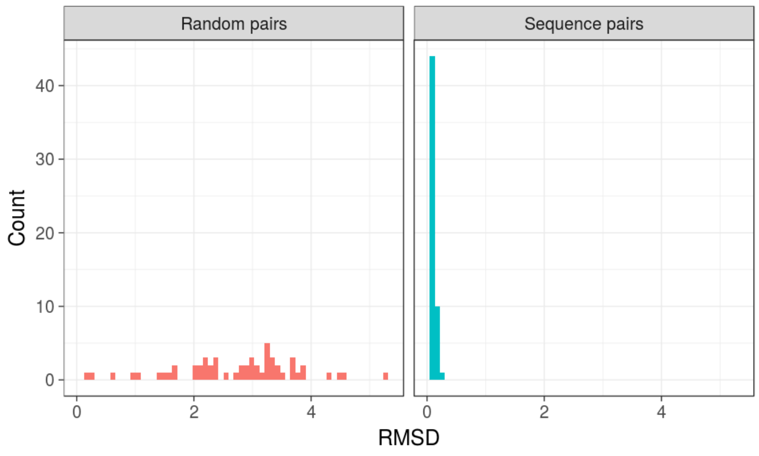


Figure 4: The 7-mers random pairs distribution spreads around a larger interval of RMSD values. The sequence pairs distribution moves toward lower values, while the random pairs become more spread out and move toward larger values. The number of n-mers decreases because matches are less probable.

3.4 Increasing the size of the n-mers promotes the divergence of the sequence pairs distribution

As a proof of concept, the whole analysis was repeated for 10-mers, that is, for pairs of aminoacidic sequences of length 7. Longer sequences increase the degrees of structural conformation freddom. This should spread out the random pairs distribution around larger RMSD values (fig 4). At the same time, only sequence pairs with low RMSD values will remain, for long sequence sharing is found only in domain families, which of course have very similar structures.

3.5 Interactive visualization of RMSD distributions

A visualization of the random and sequence pair distributions obtained for n-mers of lengths ranging from 3 to 20 can be obtained in this link:

https://antortjim.shinyapps.io/sb_exam/

4 Conclusions

This report has delved into the relationship between structure and sequence. Root Square Mean Deviation has proved to be an efficient statistic to measure the difference between 2 sets of coordinates. RMSD distributions show that short sequence sharing fragments are prone to share their structure as well. This trend is accentuaded as the length of the fragments is extended. This is opposite to what can be observed in random pairs, that show no trend to particular low RMSD values. Moreover, as the length of the fragments grows the RMSD distribution skews to larger values. This is explained by the the exponential increase in the number of degrees of freedom, that make structures diverge to a greater range of possible conformations.

Quite a lot of the observed matches came from only a few pair of proteins, and they happeneded to at least partially share their structure This shows that once a match is found between a pair of proteins, the pair becomes more likely to eventually yield more matches, due precisely to the fact that they indeed share structure at some extent. Protein comparison methods could take advantage of this property implementing n-mer seeds that heuristically accelerate algorithms.

The number of detected n-mers was extremely dependent of their length, so that only 74 6-mers are found. This suggests that further analysis with a significant amount of fragments will require larger datasets, with thousands of protein entries. For example, a filtered subset of the PDB could be valid for such an analysis.

Finally, bioinformatics methods for data analysis have proved to be very useful and effective in the performance of complex tasks. Still, Biopython does not provide a secondary structure parsing module, and DSSP, the recommended alternative, is not well integrated with the rest of the module. There is work to be done.

5 References

1. Thomas Hamelryck and Bernard Manderick. PDB file parser and structure class implemented in Python. *Bioinformatics*, 19(17):2308, 2003.
2. J M. Visualizing and Quantifying Molecular Goodness-of-Fit: Small-probe Contact Dots with Explicit Hydrogen Atoms. 1999.
3. Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, T N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.
4. W Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 32(5):922–923, sep 1976.
5. S van der Walt, S C Colbert, and G Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, 2011.
6. Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie, and Jonathan McPherson. *Shiny: Web Application Framework for R*, 2016. R package version 0.14.2.
7. Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.