University of Padova

Department of mathematics

Master Thesis in Master of Science in Data Science

A fast and effective approach for the detection of units in tandem repeat proteins

Supervisor
Damiano Piovesan
University of Padova

Master Candidate
Alberto Cocco

Academic Year

2021-2022

Abstract

Tandem repeat proteins (TRPs) are a widespread class of non globular proteins involved in many biological processes characterized by the repetition of structural and sequence elements. Mutations of the native structure of TRPs can lead to severe diseases such as the Huntigton disease or the spinal and bulbar muscular atrophy. In order to provide significance to the variants associated to those disease conditions it is crucial to provide a functional characterization of the responsible proteins, this is made possible by a thorough analysis of their sequences and structures.

Nowadays, a large amount of structural data is available in the Protein Data Bank (PDB) repository. The PDB provides structurally determined protein three-dimensional structures and it is freely available. However, despite the amount of available structural data, the majority of proteins are only known at the sequence level. Therefore, it is extremely important to identify sequence properties that can be used to transfer structural and functional knowledge starting from sequence analysis. The identification of repetitive elements, the units, inside TPRs is one of the fundamental tasks to understand the function of this class of proteins.

Despite the high structural similarity of repetitive elements inside a single TRP, the corresponding amino acids sequences are highly degenerated. Few conserved key residues are responsible for the correct folding of specific structural motifs, while the remaining positions are substituted by residues with similar physical-chemical properties and are less subjected to the pressure of the natural selection occurring during evolution and therefore more degenerated.

On the other hand, the structure of these repeated elements in TPRs are more conserved throughout evolution because their structural conformation is intrinsically associated to their function, which in turn is the real subject of the natural selection pressure. Therefore, the structure is more informative and more useful to reliably identify repeated patterns. The identification and the detection of structural repeated elements inside TRPs is often performed visually by expert biologists. However, the process is very time consuming and repetitive. Moreover, the high variability of protein structures and the complexity of their internal symmetry make the identification of the repeated elements very challenging. Also, the repeated elements often have a variable length and contain insertions and deletions.

In the last decade, a number of automatic methods to analyze the internal symmetry of TRPs have been developed. One of the most accurate is RepeatsDB-Lite a predictor able to identify repetitive units and assign a class based on the topology of the region starting from the protein structure as input. Nevertheless, the average time required to run the algorithm is about ten minutes, that makes the analysis of the entire PDB repository unpractical. The execution time is even more problematic when applied to predicted structures, such as those provided by AlphaFold which are three order of magnitude larger than the number of structures available in the PDB, totalling more than 200 million predicted structures.

In this thesis project I have developed a new algorithm that outperforms RepeatsDB-lite in terms of accuracy and is one order of magnitude faster. The algorithm performs an heuristic search against a library of repeated structural elements to identify the most likely repetition pattern, then a filtering step to optimize the position of the repetitive modules and assign the class. In the thesis I also provided an analysis of the structural library used by the algorithm in the first step and a strategy to reduce the redundancy of the examples without losing their structural representativeness.

Contents

Listing of figures

Listing of tables

Listing of acronyms

DNA	Deoxyribonucleic acid
PDB	Protein data bank
RMSD	Root mean square deviation
RNA	Ribonucleic acid
TM-score	Template modelling score
TRP	Tandem repeat proteins
SRUL	Structural repeat unit library

1 Introduction

This chapter main aim is to give a brief biological introduction and to explain the motivations of this work. First i will give a short introduction about proteins, the main building blocks of life. After that i will explain why the process of folding is so essential to make most of the protein functional and i will show some examples of structural motifs that are common in many polypeptides. Then i will shift the focus on non-globular proteins. in particular tandem repeats proteins and their classification in RepeatsDB. Finally i will explain how we can compare structures and i will show some metrics that are usually used in the literature to establish quantitatively the degree of similarity between two structures.

1.1 Aminoacids

Before going through a brief biological description of proteins, it's worth to define what is an aminoacid, the main component of a protein sequence.In nature more than hundred aminoacids exist but only 20 of them appear in the genetic code: each of them has some physical/chemical properties that are responsible for determining the final structure of the protein. Each aminoacid consist of an α-carbon to which other components are attached as depicted in figure 1.1:

- A hydrogen atom
- A basic amino group
- An acid carboxyl group (-COOH)

- A R group which is variable for each aminoacid providing peculiar physical/chemical properties.

Aminoacids are linked together during protein synthesis occuring in ribosomes and they are usually divided in two broad categories:

- Hydrophobic aminoacids have a very low propensity to get in contact with water
- Polar or charged aminoacids that are usually more hydrophilic.

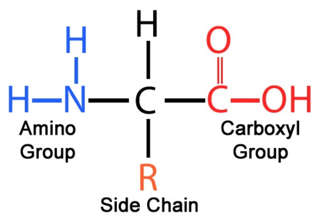

Figure 1.1: Aminoacids general structure

Those properties define some common behaviours taken in protein folding: aspartate, arginine and glutamate which are strongly hydrophobic and charged aminoacids have a very high probability of having their surfaces exposed to the solvent, thus they will be found in coiled regions and in loops where they will give more flexibility to the protein chain. Other aminoacids, instead, such as proline are hydrophobic, thus they have a higher probability of being buried in the protein core even if loop regions of α-helices are often enriched in proline aminoacids.

1.2 Proteins

Proteins are essential biomolecules composed of monomers called aminoacids linked by strong peptide bonds. They are involved in almost all the biochemical reactions carried out by our organism and they are called polypeptides when they are made of less than 100 aminoacids, oligopeptides when they are very small and composed of just 20-30 aminoacids and proteins when they are instead made of more than 100 aminaocids. Often proteins are organized in long proteins chains having a unique aminoacid sequence also called primary structure. The primary structure is usually represented as a sequence of characters in which each character encodes a different aminoacid determining the final folded structure of the molecule. They can assume three fundamental types of function:

- **Structural**: they mantain the structure of cells and organs. An example is collagen representing 25% to 35% of our body whole proteome
- **Catalytic**: they are involved in many biological processes speeding up chemical reactions
- **Transport**: given the hydrophobicity of the phospholipid bilayer they can interact with substances outside the cell membrane helping them to pass and get inside.

Usually proteins are synthesized by our organism in organelles located inside the cell and called ribosomes. The DNA biomolecule is made of a sequence of molecules called nucleotides: cytosine (C), guanine (G), adenine (A) and thymine (T). Each subsequence of DNA involved in encoding for a protein is known as gene. Genes are first transcribed into pre-messenger RNA inside the nucleus of the cell by a specific protein called RNA polymerase and some specific transcription factors. Before going through translation, the pre-mRNA is still processed inside the nucleus undergoing to different post-translational modifications (splicing, 5' cap and poly-A addition) which are essential to obtain a stable and mature RNA. At this point, the mRNA migrates in the ribosomes where the actual translation of RNA takes places. The mRNA is read in triplet called codons encoding for a specific aminoacid: each codon is then matched with an anticodon located on a trasfer RNA molecule carrying a specific aminoacid which will become a new building block for the newly translated polypeptide.

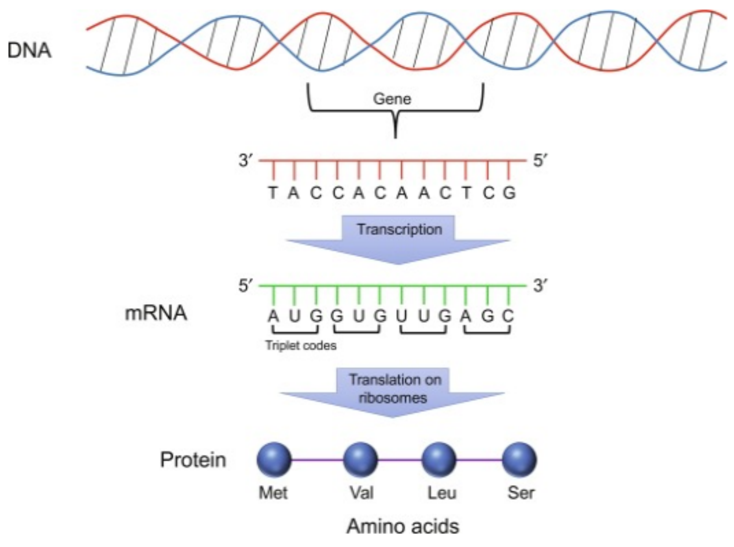

Figure 1.2: The central dogma of molecular biology

1.3 Principles of protein folding

The linear aminoacid unfunctional sequence produced by the ribosome is called primary structure because it determines the native conformation the peptide will take. The specif aminoacid sequences (not the aminoacid composition itself) encodes for the final three-dimensional structure of the protein and the whole pathway necessary to achieve that state. As a consequence the folding pathway of two identical protein sequences will be very similar even if sometimes some environmental factors can modify the process. Folding is a spontaneous process taking place in all proteins either during the biosynthesis of the protein by ribosomes of after the synthesis of the polypetide is completed, driven by hydrophobic interactions where intramolecular forces like Van Der Walls forces and hydrogen bonds are fundamental actors. Other factors can influence the process such as ph, salt concentration and even other proteins called chaperones involved in checking that conformational folding occurs correctly. The most important driving force in the whole pathway is called the hydrophobic effect. Hydrophobic effect makes hydrophobic chains collapsing in the core of the protein: solvent molecules tend to aggregate around the hydrophobic region increasing the order in the system. The solvent molecules are fixed in these cages causing the collapse of hydrophobic chains. The collapse contributes breaking water shells freeing water molecules. The multitude of hydrophobic chains interacting with the core of the protein increases by a lot the stability of the system during folding because of the high amount of London dispersion forces accumulated.

1.3.1 Secondary structure

Formation of the secondary structure is the first step towards the final native conformation of the protein. Most of the possible arrangements of adjacent residues fall into four main classes determined by the ϕ(phi) and ψ(psi) angles formed between the peptide bonds and the α-carbon, obtained by rotating two adjoining plains connecting to the carbon itself. As a consequence the local secondary structure elements taken by adjacent residues is determined by the set of possible ϕ(phi) and ψ(psi) angles. In this regard ramachandran plots are great graphs to show the propensity of adjacent residues to form a specific secondary structure based on dihedral angles.

- α-helices are right-hand helix conformations in which every backbone N-H group donates a hydrogen bond to the backbone C=O group located three or four residues earlier in the protein sequence. The hydrogen bonds linking distant residues in the protein sequence provide a special stability to those type of structures.

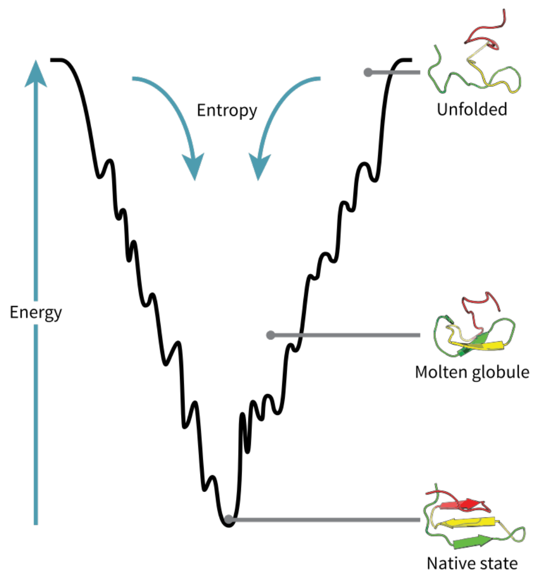

Figure 1.3: folding process can be represented as a funnel: a number of intermediate pathways and states is present and directed toward the native state. The folding process is spontaneous and tries to minimize the free energy of the system and the entropy

- β-sheets consist of beta-strands connected by at least two backbone hydrogen bonds forming generally a pleated or twisted sheet. The assembly of β-sheets can be parallel or antiparallel depending on the orientations of the beta-strands.

Other secondary structure elements are present in nature and they can be easily predicted by some software. However, it's important to recall the molecular shape and size of aminoacids is a fundamental constraint in the type of dihedral angles they can form: thus specific aminoacids have preferences for some secondary structure conformations confirmed also by advanced statistical analysis techniques.

1.3.2 Tertiary structure

During the formation of the tertiary structure secondary structure elements can bond together forming more stabilizing and energy favourable three-dimensional arrangements. Secondary structure elements are made of hydrophilic and hydrophobic portions. For this reason folding usually proceeds in a way that hydrophilic residues can face the solvent surrounding the protein while hydrophobic residues are called "buried residues" hence they become part of the protein core. β-sheets are involved in the burial of non polar residues, while α-helices have a higher flexibility and they may contribute in the exposure of hydrophilic residues and in the internalization of hydrophobic residues.

1.3.3 Quaternary structure

Tertiary structure is often not the final conformation polypeptide can take. During the formation of the tertiary structure polypeptide chain folds indipendently from each other. In many cases proteins are made of more than one chain that can bond together to determine the final and functional protein complex. More precisely, not only chains but also different functional subunits (domains) can fold independently and then assemble to give origin to a macromolecular complex during the formation of the quaternary structure. This complex process improves significantly the existent biological variability: when the two subunits are identical they can form a complex called homodimer. On the other hand when the two subunits are different they can form an eterodimer. Furthermore when more identical units link together they can form a polymer.

1.4 Experimental determination of protein structures

The formation of the tertiary structure is essential to make a clear distinction between two groups of proteins:

- Globular proteins have a rounded and compact structure where hydrophilic residues are exposed to the solvent and hydrophobic ones are located in the core. They have multiple functions in metabolic pathways
- Non-globular also called intrinsically disordered or fibrous proteins have an elongated shape and they are often not fully folded. They can be divided in further subcategories based on their morphology which will be explained more precisely in 1.6

Determining the function and the interactions a protein can assume is essential: therefore different procedures coming from biophysics were exploited. X-ray cristallography is a high-throughput technique based on the crystallization of protein structures. The crystal is illuminated using a X-ray monochromatic beam that produces a diffraction pattern in many different directions. A goniometer is used to position the crystal in many different directions. The multiple images obtained by varying the orientation of the crystal are then processed using fourier transformations to achieve a three-dimensional model. This method is very convenient for globular proteins, but it fails when it's more difficult to retrieve a crystal structure as in the case of intrinsically disordered proteins. Moreover, another problem is the time required to get the crystal that can take also months. In this regard another method very used is the nuclear magnetic resonance (NMR) spettroscopy based on the observation of local magnetic fields around atom nuclei. The sample is placed in a magnetic field and the NMR signal is produced by the excitation of nuclei sample with radio waves. The intramolecular magnetic field around an atom in a molecule changes the resonance frequency giving access to the details of the electronic structure. Structure retrieved from experimental procedures are then stored in the Protein Data Bank database (PDB)* where they are freely available for download. A total of 192888 structures is available in the Protein Data Bank database at the moment of writing.

1.5 Non globular proteins

The world of proteins can be divided in two big families: globular structures and non-globular or fibrous structures. Globular structures have a spherical and compact shape and they are usually water soluble. Non globular proteins, on the other hand, have a elongated shape and they usually lack a fixed and stable three-dimensional structure. They played an important role in disproving the idea that a protein has to maintain a fixed three-dimensional structure to be considered functional. Indeed the first protein structures were solved in 1930-1950 by X-ray crystallography suggesting that a fixed three-dimensional structure might be necessary for a functional protein. During the subsequent decades many large protein regions could not be assigned to an X-ray datasets indicating that they can occupy multiple positions in the electronic maps. The lack of fixed, unique positions relative to the crystal lattice suggested the presence of disordered domains and regions. Today, in 2022 a growing evidence has revealed the involvement of non-globular proteins in many biological processes and diseases. In particular it was estimated that 40 % of the human proteome lacks a functional annotation and many of those proteins contain repetitive or disordered regions [1]. At the moment, for as far as we know we can represent the world protein as depicted in 1.4. By first looking at the left part of the picture two types of full non-globular structures are shown. **Intrinsically disordered proteins** are the most known and they usually lack a three-dimensional structure when they are not interacting with other partners (DNA, RNA or other proteins). **Aggregates** are instead the consequence of a biological phenomenon called aggregation occurring inside or outside the cell in which partially structured agglomerates are responsible for diseases such

*PDB database: https://www.rcsb.org/

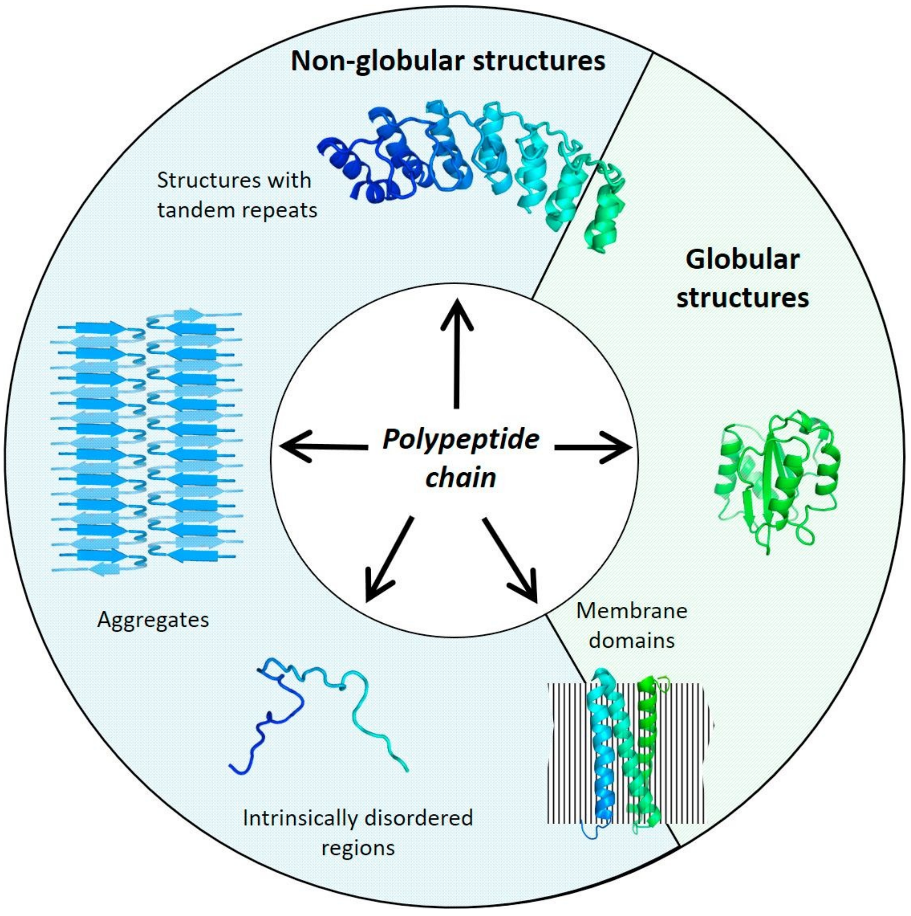

Figure 1.4: Classification of proteins according to Andrey V. Kajava and Silvio C.E. Tosatto. Image taken from Andrey V.Kajava, Silvio C.E. Tosatto, Editorial for special issues: "Proteins with tandem repeats: sequences, structures and functions ", Journal of Structural Biology (2018)

as amyloidoses, Alzheimer's or Parkinson's. The formation of these aggregates has different causes: aging, for example, can weaken cell proteins able to degradate and remove agglomerate from the cell or mutations caused by environmental or genetic factors can interfere with the process of transcription or translation of RNA resulting in a partially different protein sequence that can completely change the process of protein folding leading to a partially or totally unfunctional molecule. **Transmembrane proteins**, instead, are in the half between a globular and a non-globular structure: the central part of the protein is stable and embedded in the cell phospholipid membrane while the N-terminus or the C-terminus are more flexible and they are often located in the outer part or in the cytosolic side of the cell undergoing frequent conformational changes due to their importance in passive and active transport. Last but not least for importance are Tandem repeat proteins which will be the focus for the next chapter.

1.6 Tandem repeats proteins

Tandem repeats proteins (TRPs) are proteins having one or more adjacent copies of the same structural motif that can expose a domain. They cannot be considered fully globular proteins since most of them have not a compact and rounded shape, but at the same time they are not even fully disordered given the presence of regular repetitive motifs that make the structure very stable. Even if tandem repeats proteins maintain a high structural similarity, they can be highly degenerate at the level of the sequence with only a few aminoacids conserved. As stated by Andrey V. Kalaja [2] they are ubiquitous in genomes and they occur in 14 % of all proteins. Moreover, they make 1/3 of the human genome. They are very adaptive but characterized by a strong genetic instability, thus the deletion or the insertion of a new unit can rarely lead to a complete misfolding of the protein. Indeed the strong genetic instability is caused by two main mechanisms underlying in our genome: one is replication slippage occuring in DNA replication where some complementary bases are misplaced during DNA replication and the other is DNA recombination which takes place during meiosis and it's responsible for random exchanges of genetic materials . They can extremely different shapes: some of them are closed and made of long units (more than 40 residues), some others are elongated and made of just 15-20 residues.

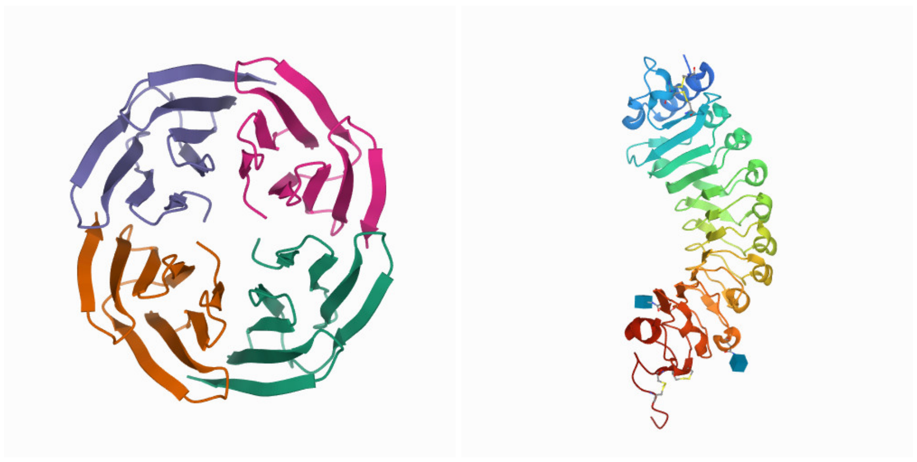

Figure 1.5: On the left PDB 6r5x is shown and it's a typical closed repeat structure called propeller. On the right PDB 6w78 is an elongated structure called solenoid

1.6.1 Defining tandem repeat proteins

Tandem repeats proteins are defined by three elements:

- **Region** is the part of the protein that contain the repeated elements. A protein can be made of multiple regions
- **Unit** is the smallest structural block that keeps repeating inside a region
- **Insertions** are positions that escape the definition of the units for the considered region. They are usually located inside a unit or between units

In order to distinguish consecutive units, two colours are used (red and blue). Yellow, instead, is used to highlight insertions. Figure 1.6 and 1.7 show two examples.

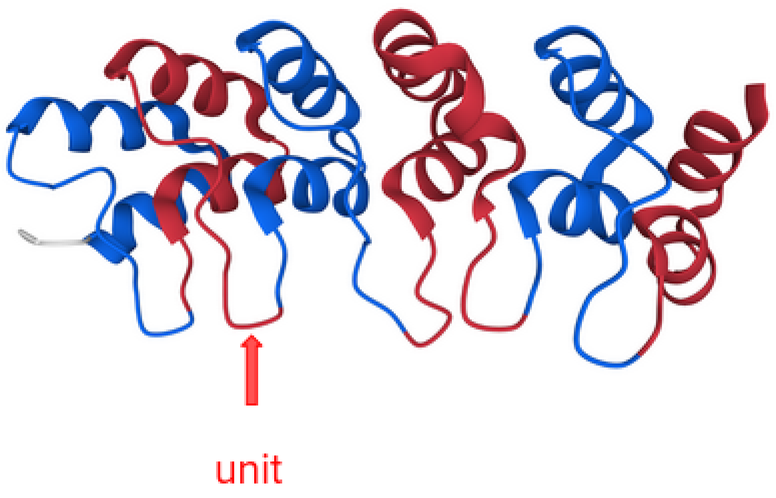

Figure 1.6: Example of an annotated region of a PDB: units are coloured in red and blue

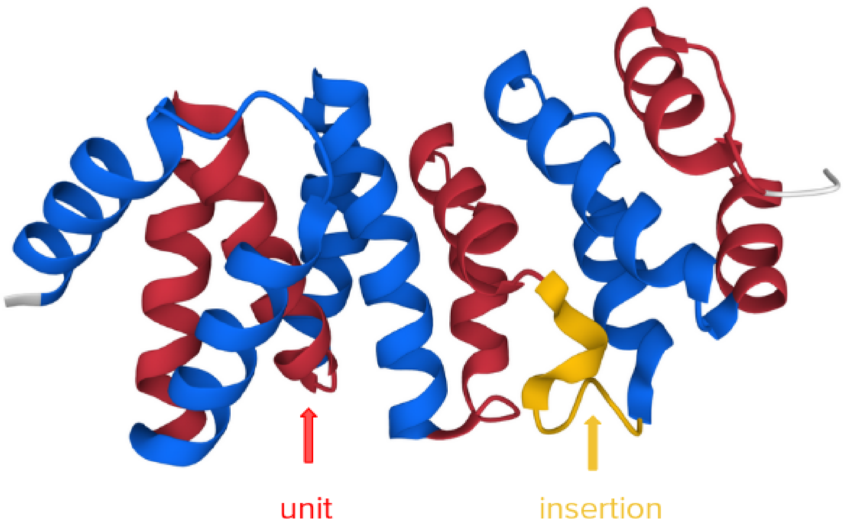

Figure 1.7: Example of annotated region containing one insertion

1.6.2 Why deepen our knowledge on repeats ?

One of the question that can arise at this point is the reason why tandem repeats should be studied in details and why scientist are interested in building a formal and precise classification for them. The motivation is pretty straightforward and it can be easily found in a Andrey V. Kajava paper [2]: Beside the fact that they are very frequent in serious neurodegenerative diseases (e.g. Parkinson's, Huntigton) the main motivation is because many of them are part of fundamental protein interaction networks and cell transport systems. Even though the majority of repeats is found in eukaryotes, many viruses and bacteria have tandem repeat regions in their proteins sequences characterized by a high genetic instability that under specific conditions can be a major threat for our health. Furthermore tandem repeats proteins are very studied to develop also new vaccines able to detect very specific epitopes located on target antigens of pathogens.

1.6.3 RepeatsDB

Even if the amount of available structures is not even comparable to the number of available protein sequences (UniProt)[†], new structures are inserted in the PDB database every day. Beside that, right now there are very advanced bioinformatics algorithms like AlphaFold that are able to predict with a high accuracy the secondary/tertiary structure of most of the newly dis-

[†]UniProt database: https://www.uniprot.org/

covered sequences. From those two points it's becomes clear we need a systematic way to store and classify the high amount of structures that every day are discovered, but the PDB database is not sufficient. In fact proteins are structurally and functionally very different and having a unique classification for both globular and non-globular proteins is definitely not ideal. Even if we try to build a standard unique classification for non-globular protein we would probably not succeed: intrinsically disordered and tandem repeats proteins are structurally very different and need to be annotated in specific and appropriate ways. For our discussion we are mainly interested in RepeatsDB[‡] [3] a database of manually and automatically annotated tandem repeat proteins where each entry is a PDB chain. Even if only a representative part of the database is manually annotated by expert biologists and the rest of the entries is automatically curated using the sequence and structural predictor RepeatsDB-Lite (see 1.8.1), the majority of UniProt has at least one corresponding structure manually annotated as highlighted in pie chart 1.8. The process of annotation usually expects three different steps that are not necessarily executed in the presented order:

- Definition of the start and the end of the repetitive region
- Definition of the units boundaries inside the region and eventually annotation of the insertions between or within units
- Classification of the region based on four different levels which will be further explained (class, topology, fold and clan).

[‡]RepeatsDB:https://repeatsdb.bio.unipd.it/

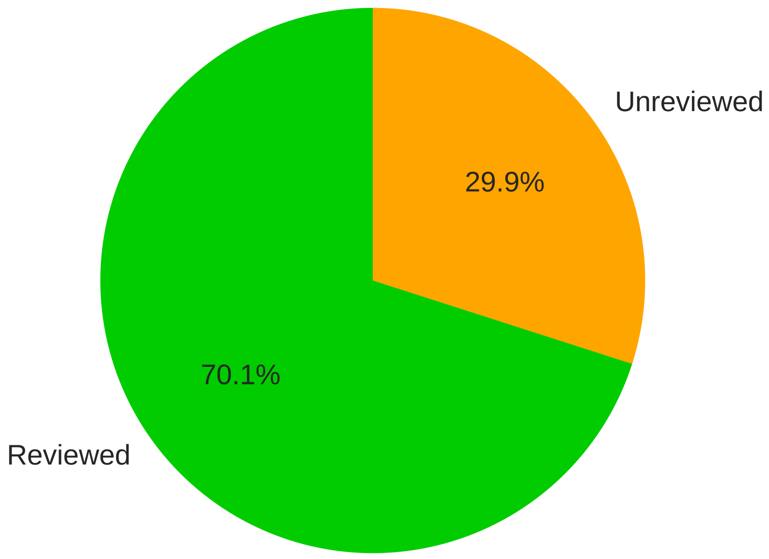

Figure 1.8: The pie chart shows the number of UniProt for which at least one PDB is manually annotated in RepeatsDB (June 2022). Only the 29 % of the total number UniProts (1513) has no structures reviewed.

It's trivial to point out that manual curation is a crucial and a challenging task to carry on. Structures are very different and sometimes single units or entire regions could undergo different evolutionary processes making difficult to detect precisely insertions and units. For this reason discussion is vital and it is mandatory to define a standard protocol that can be consulted by all curators. The classification used by the database is based on the work of Andrey Kajava[2] and it is built on five levels described here from the outer to the inner level:

- **Class** is the outer level. It depend on the repeat length and the generic interaction between repetitive elements inside the region. Often very easy to define. RepeatsDB classes are shown in figure 1.9
- **Topology** is determined by the type of secondary structure taken by the units in the repetitive region.

- **Fold** level of classification dictated by the overall arrangement of secondary units (twists and curves)
- **Clan** A subfold that groups up proteins having common sequence motifs that are part of the repeat but not a common ancestor. Many of the structures in RepeatsDB have not a clan definition.
- **Family** Group of proteins having a common ancestor and a high degree of sequence similarity

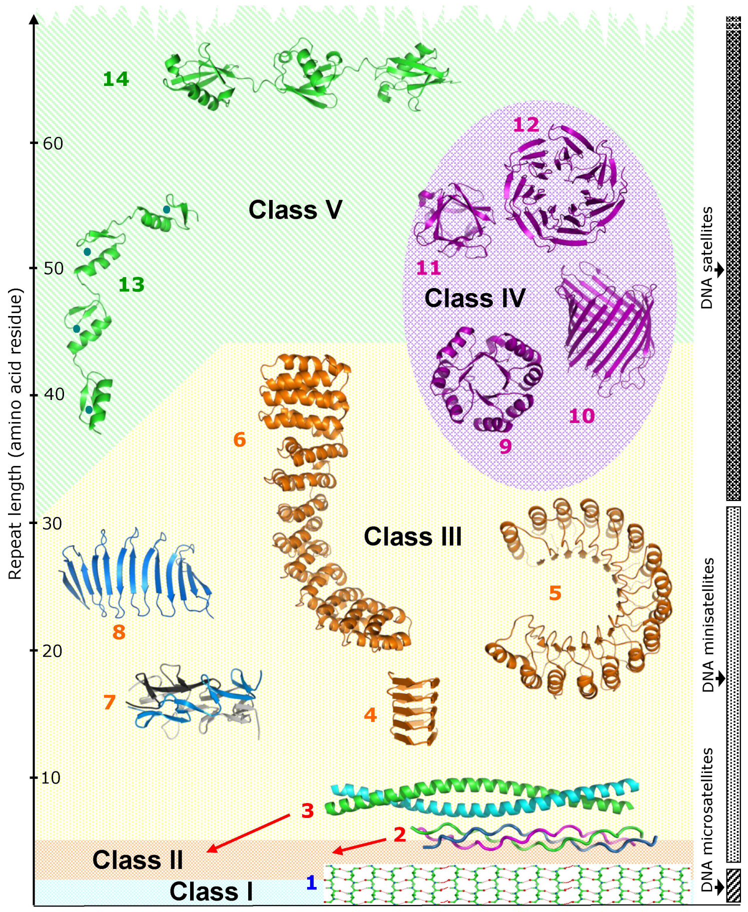

Figure 1.9: The classification of repeats based on repeat length. Structures having shortest repeats are crystalline aggregates underrepresented in RepeatsDB (Class I). Class II are fibrous repeats having a slightly longed repeat length. Class III are elongated repeats. Class IV are closed repeats the most present in RepeatsDB. Class V are beads-on-a-string the ones having the longest unit length. The image was taken from ”Tandem repeats in proteins: From sequence to structure”, Journal of Structural Biology, Andrey V. Kajava (2012)

1.7 Comparing structures

One of the main goals of structural bioinformatics is to infer knowledge about new structures from homologous proteins. Two main approaches are well known in the literature: one is based on alignments and the other on superposition. Structural alignments require no a-priori knowledge and can be used to compare structures having a variable length. Given two sequences and structures as input a structural alignment algorithm can compute the optimal alignment between the two structures Superposition, on the other hand, does not need the sequence but requires a set of aligned coordinates of the atoms to find the best rotations and traslations that can minimize the difference between the two structures.

1.7.1 Structural alignments

Structural alignments approaches do not require any previous knowledge. They are useful when no sequence homology can be detected just by sequences using multiple sequence alignments due to low sequence similarity (structures tend to be more conserved). Structural alignments tools give as output the set of aligned coordinates from which a bunch of metrics to evaluate the quality of the alignment can be derived (RMSD, TMscore and GDT). The main issue with this type of approaches is time complexity: finding the best alignment is a NP-hard problem because to align N residues from a structure to M segments a total of N^M combinations has to be produced. As a consequence most of the methods known in the literature are just heuristics able to approximate the optimal solution. From the literature one of the most known methods is DALI[4] a common method that uses distance-matrices to evaluate the contact patterns between successive fragments of the input protein. Combinatorial extension[5], similarly to DALI, break the input structure into fragments which are used to generate a distance matrix from which the optimal alignment is derived. SSAP[6] use dynamic programming to compute the distance between interatomic vectors and instead of considering α-carbons it takes in consideration β-carbon. However the most known and used tool nowdays is TM-align[7]: TM-align generates residue to residue optimal alignments through multiple iteration of dynamic programming using a TM-score matrix. The main difference with other methods is that it does not evaluate the quality of the alignment based on RMSD but it uses a metric called TMscore which will be further explained in section 1.7.4. The software can be used from the web interface available at `https://zhanggroup.org/TM-align/`.

1.7.2 Superposition

Superposition is way simpler than structural alignment and it is based on geometrical transformations, but does require previous knowledge on our data. The two sets of data that need to be compared have to be of the same size and equivalent, thus it usually used to when comparing different conformations of the same protein. The basic and most known algorithm is the Kabsch algorithm in which a simple least squares method for convex optimization find the best rotation and traslation matrices that minimize a target function (the root mean square deviation) between the two sets of atoms (see `https://en.wikipedia.org/wiki/Kabsch_algorithm`).

1.7.3 Root mean square deviation

The root mean square deviation is the average distance between the atoms in equivalent positions after applying superposition. To make the computation less heavy it is a common practice to consider only α-carbons. Two main problems are the fact that is very sensitive to local deviations in the structure even when the overall shape and topology of the two structures is nearly identical and the fact that it's a non-normalized quantity. Usually an RMSD lower than 2 define two structures that share a strong similarity, but in general it strictly depends on the type of structures we are considering.

$$RMSD = \sqrt{\frac{\sum(r_{ai} - r_{bi})^2}{n}} \tag{1.1}$$

where

- r_{ai} and r_{bi} are the coordinates of the i equivalent atoms in structures a and b
- n is the number of paired atoms in the structure.

1.7.4 Template modelling score

The template-modelling score, as the RMSD is a measure of similarity between two protein structures. It is a more accurate for evaluating the global similarity between two protein and it is bounded between 0 and 1 where 1 indicates a perfect match between two protein structures.

Scores below 0.20 correspond to unrelated structures, while scores greater than 0.5 are a good sign of some shared generical similarities. The formula for the TM-score is:

$$TM - score = \frac{1}{L_{target}} \sum_{i}^{L_{common}} \frac{1}{1 + \frac{d_i}{d_0(L_{target}})} \qquad (1.2)$$
$$d_0(L_{target}) = 1.24\sqrt[3]{L_target - 15} - 1.8$$

where:

- L_{target} is the length of the aminoacid sequence of the targer protein.
- L_{common} is the number of residues that appear both in the template and the target structure
- $d_0 L_{target}$ is a distance used to normalize quantities

The d_i in the denominator is crucial because it gives more weight to large distance errors while it weights smaller short distance errors making it ideal for detecting global similarities.

1.8 Meaningful approaches from the literature

Making a fast and efficient predictors for unit prediction and region annotation of tandem repeats proteins is not piece of cake: many efforts and very different approaches were explored in the past years. First attempts were made around 1999 using sequence based methods since the number of structures available was not that meaningful to develop a structure-based predictor. Heger and Holm[8], in 2000, developed RADAR a method based on finding sub-optimal alignments in the self alignment matrix using the smith-waterman algorithm for pairwise alignments. Another algorithm, developed in in 2007 was T-REKS[9] using a clustering approach based on K-means. REPETITA [10], instead, was able to make predictions on solenoidal structures applying discrete-fourier transforms from biochemical features (polarity, secondary structure, codon diversity, molecular volume etc..). From the first years of the XXI century different structure based methods were implemented: indeed it has been proved that tandem repeats proteins tend to share structural motifs but not always a strong similarity, hence methods based on the detection of repeated elements from the primary sequence can fail even in simple cases. Structure based methods are able to detect repetitive elements using the three-dimensional coordinates of residues atoms (mainly α-carbons) retrieved from experimental methods. The only problem is related to the range of applicability of those methods since the number of available structures is extremely lower compared to the number of sequences for different reasons. As a consequence many methods focus only on some specific kind of structural motifs. RAPHAEL[11] was one of the most successful algorithms for the detection of solenoids in protein structures: the input protein is subjected to traslations and rotations and for each of the three three-dimensional coordinates of the α-carbon a projection is performed. Another method called TAPO[12] uses periodicity of atomic structures, distance matrices, contact maps and vectors of secondary structure elements to predict unit displacement in the repetitive region. One of the most successful approaches was ReUPred[13], an algorithm for the detection of solenoidal repeats based on structural alignments between the input structure and a library unit which was called for the first time SRUL. In the next years the algorithm was updated in order to be able to annotate proteins belonging to all classes of RepeatsDB and the new release of ReUPred has been called [14]RepeatsDB-Lite. RepeatsDB-Lite is the algorithm currently used by RepeatsDB to predict units and annotate repetitive regions which will be further explained in the next section.

1.8.1 RepeatsDB-Lite

RepeatsDB-Lite is the automatic predictor used to identify units and annotate tandem repeats regions of the tandem repeats proteins that are contained in RepeatsDB. The algorithm is based on a divide and conquer approach and uses both structural and sequence related information. The required inputs for the algorithm are the Structural Repeats Unit Library and a FASTA sequence or a PDB structure. If a sequence is provided the algorithm searches in the PDB database for the corrisponding structure, otherwise the most similar one. The first step of the algorithm structurally aligns each unit of the SRUL with the input structure using the external software TM-align. The best matching unit is the one with higher TM-score and lowest RMSD and it is called the master unit. From now on the algorithm uses a recursive approach and splits the input structure in two fragments (PRE and POST) corresponding to the N and C terminals flanking fragments of the first predicted unit. The cycle is repeated on those two segments until the length of the two fragments is less than 85 % of the length of the first predicted unit. Once the alignment and identification of units has finished the units are collected together in regions and gaps longer than 4 residues are labelled as insertions. Each predicted region inherits the class and topology annotation given to the master unit. The last step of the algorithm identify insertions inside units: it uses MUSTANG, a software for structural alignments, to produce a multiple sequence alignment and determine the columns where there are more than 85 % of the gaps. RepeatsDB-Lite is certainly a well performing algorithm with a complex but well studied

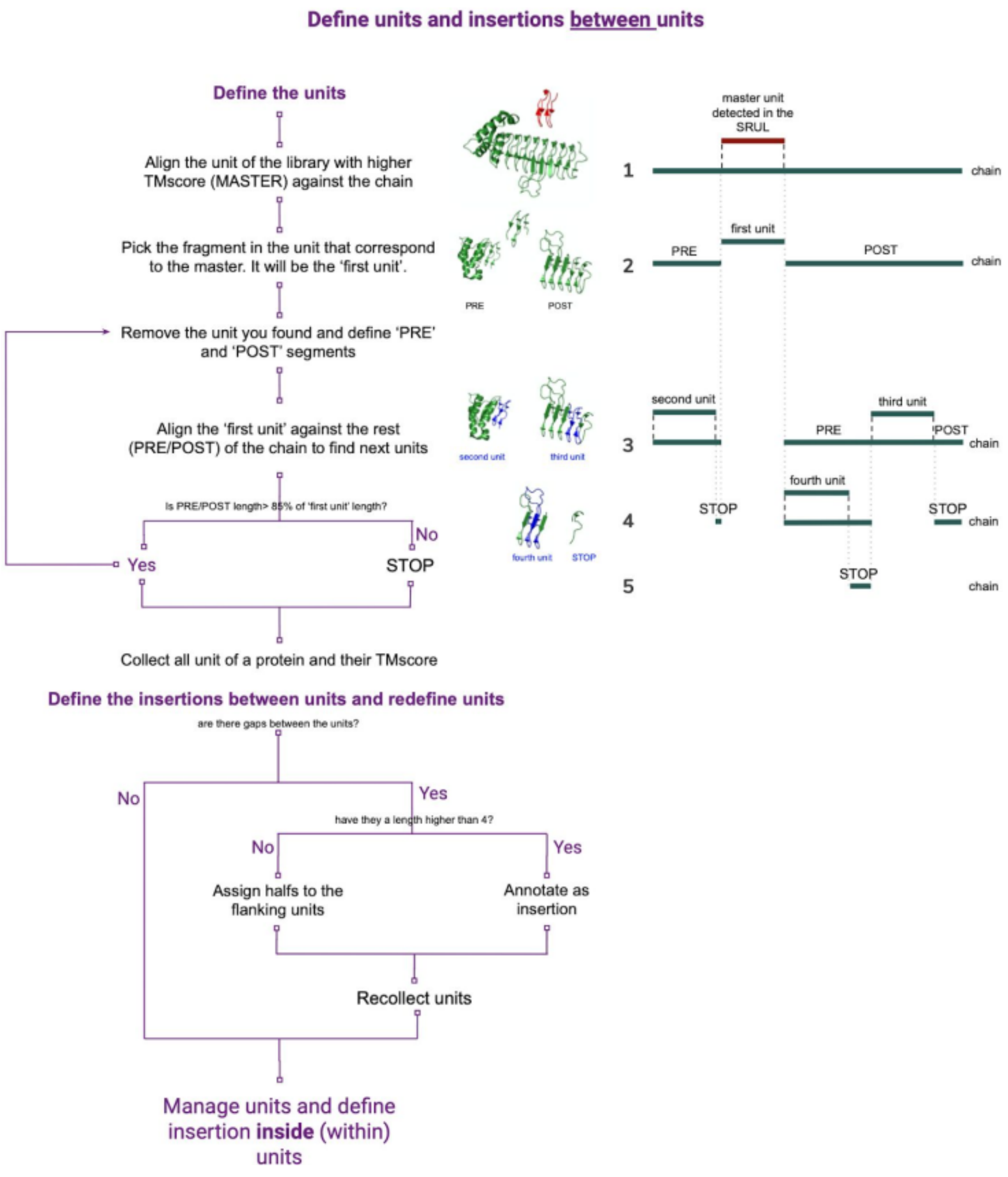
Define units and insertions between units
Define the units
Align the unit of the library with higher TMscore (MASTER) against the chain
Pick the fragment in the unit that correspond to the master. It will be the 'first unit'.
Remove the unit you found and define 'PRE' and 'POST' segments
Align the 'first unit' against the rest (PRE/POST) of the chain to find next units
Is PRE/POST length> 85% of 'first unit' length?
Yes
No
STOP
Collect all unit of a protein and their TMscore
Define the insertions between units and redefine units
are there gaps between the units?
No
Yes
have they a length higher than 4?
No
Yes
Assign halfs to the flanking units
Annotate as insertion
Recollect units
Manage units and define insertion inside (within) units
master unit detected in the SRUL
1
chain
2
first unit
PRE
POST
chain
3
second unit
third unit
PRE
POST
chain
second unit
third unit
4
fourth unit
STOP
STOP
chain
fourth unit
STOP
5
STOP
chain

Figure 1.10: schematic representation of the RepeatsDB-Lite algorithm

structure that mimics the evolutionary process of folding proteins. However the algorithm has some issues:

- On average it requires from 10 to 15 minutes to run. Clearly, the algorithm is too slow if we want to use it for the increasing number of structures available each day.

- It relies on external software like MUSTANG or TM-align. MUSTANG often crashes and alignments themselves can slow down the process.
- Sequence information are not fundamental. It is possible to build a good predictor even by just using structural features. Probably we sacrifice some precision in recognizing insertions but in general it is something very difficult to estimate: how can we establish the difference between a small gaps and an insertions or if a long insertion is just a unit which was not predicted?
- RepeatsDB-Lite is not always able to detect repetitive elements on long input stuctures. Often Alphafold predictions need to be cut off to be processed.

Summarizing the overall purpose is to design a faster algorithm able to process also long structures in a feasible amount of time. The new algorithm, which will be further explained in chapter 4, is a structural predictor that detect units by superimposing unit fragments in the SRUL with fragments of the input structure.

1.9 Thesis outline

The thesis is organized as follows. In Chapter 2 I have explained how I have generated the data for the Structural Repeat Unit Library (SRUL) including information about the data source, formats and filtering procedures. Chapter 3 is focused on the analysis of the SRUL and on how hierarchical clustering can be used to reduce its redundancy without losing information. Chapter 4 provides a complete description of the new algorithm that I have called RepeatsDB-Lite version 2. Also, I have provided a comparison with the state of the art and an analysis of the effect of the SRUL clustering on the accuracy and speed of the new algorithm. Conclusion remarks are reported in chapter 5.

2
The structural repeat unit library (SRUL)

The main aim of this chapter is to explain how the data coming from RepeatsDB were used to build the so called Structural Repeat Unit Library (SRUL). In the first sections i will make some exploratory data analysis and i will show some statistics about the data i used from RepeatsDB. Then i will explain how i filtered the data exploiting sequence information and how i generated the SRUL.

2.1 What is the SRUL ?

The Structural Repeat Unit Library is a library that, in theory, should represent the main and most generic unit conformations that is possible to find in nature. Right now, it is almost impossible to build a perfect SRUL for three main reasons:

- The number of UniProts entries that are currently available in RepeatsDB (1511) is still not representative
- Regions that have been annotated are mainly propeller, alpha solenoids and TIM-barrels. Other classes, for example beads-on-a-string have an extremely lower number of representatives in respect to closed and elongated repeats
- How we can decide what is the ideal number of units we should store in the SRUL for each class ? This is another crucial question that has no answer. Probably only empirical approaches would be helpful in this regard.

Taking into account what have been said above, it is clear that building an optimal SRUL is a high effort and long time process that can be gradually achieved having an increased amount of data, in particular for classes that are right now less represented. On that note our main goal is just to build a good and updatable library with the data available even if it will be biased, with very high probability, towards specific kind of units.

2.2 RepeatsDB data

From 2021 on, after a common standard for manual biocuration has been defined, RepeatsDB is going through a continuous process of update and new structures are annotated every day. The data i will consider for this project are all the repeated regions that were new annotations or recently reviewed until May 2022. In total i had 17352 units coming from 2119 PDB chains and belonging to 2285 regions. The number of Uniprots considered was 917 which is not far lower than the total amount of UniProts that are currently conserved in RepeatsDB (1511). As depicted in graph 2.1 the distribution of units between classes is not balanced: Closed repeats and Elongated repeats have a quite similar amount of units but compared to Beads-on-a-string and Fibrous repeats the difference is huge. Fibrous repeats are just 3, while Beads-on-a-string are extremely underrepresented. Crystalline aggregates were not even taken into account since they are not even available in RepeatsDB. If we compare these data with what there is right now in RepeatsDB we can verify that proportions are more or less maintained: a total of 43415 Elongated repeats and 55042 Closed repeats are available against 393 units for Fibrous repeats and 2091 for Beads-on-a-string. In percentage 47.9 % of the units belonging to the units in the dataset are elongated repeats against the 43.01 % of RepeatsDB, 49.9 % are closed repeats against the 54.5 % of RepeatsDB and 2 % are beads-on-a-string against the 2 % of RepeatsDB. A reliable difference can be seen only in Fibrous repeats even if they make up only 0.4 % of the total number of units in RepeatsDB. Taking a look at the distribution of the topologies (bar plot 2.2), instead, we can retrieve other interesting information. All classes, except for Beads-on-a-string, are biased towards specific topologies probably due to the large abundancy of these motifs in nature. Concerning Elongated repeats we can notice Alpha-solenoids constitute a vast majority: these structures are made of repeating alpha-helix subunits of 30-45 aminoacids that can be found very frequently in genomes even if frequencies are variable: they are more rare in prokaryotes and very frequent in eukaryotes. They have different function but most of them are present in protein interaction networks of regulatory proteins and in nuclear pores proteins complexes. Concerning closed repeats, instead, we can notice how TIM-barrel and propellers

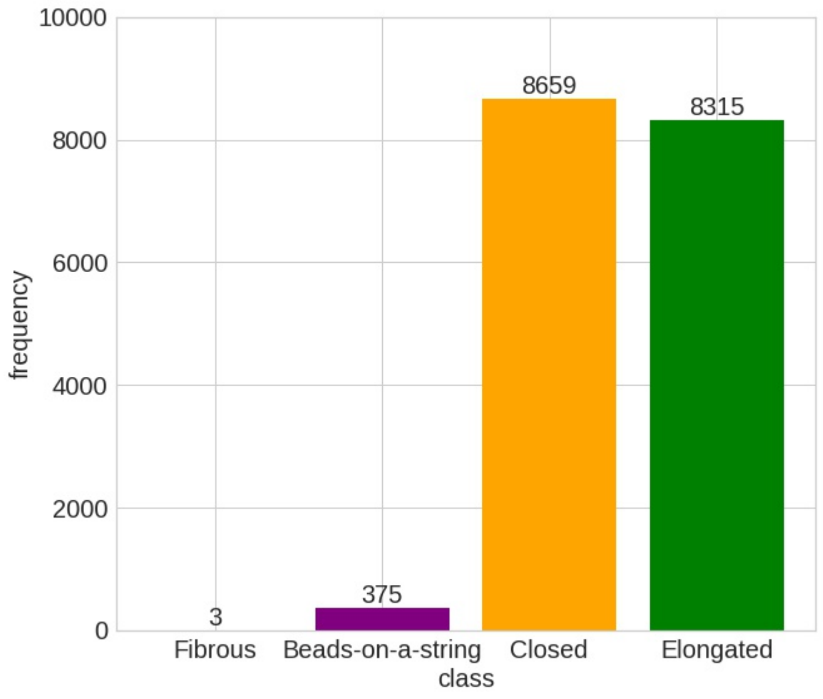

Figure 2.1: Barplot highlighting the distribution of units between the 4 classes

are both very present. TIM-barrels are weaved structures made of eight alpha-helices and eight beta-strands alternating on the protein chain. they are very ubiquitous in enzymes since 10 % of enzymes takes this fold. Moreover five of seven enzyme commission classes include TIM-barrel proteins. On the other hand, propellers, also called beta-propellers are all beta protein structures characterized by 4 to 8 highly symmetrical blade-shaped beta sheets arranged around a central axis. They are widely spread in nature because they are often involved in ligand binding and enzymatic activity.

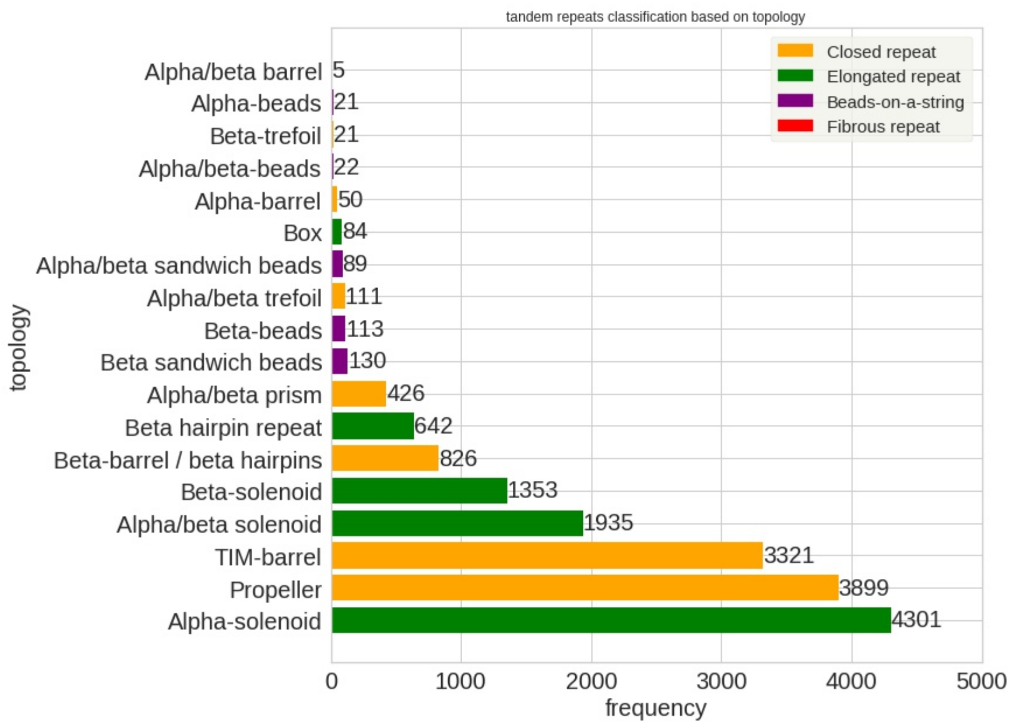

Figure 2.2: Barplot showing the distribution of units among the topologies in the dataset. Alpha-solenoids are a vast majority for elongated repeats, while Propellers and TIM-barrel are a very abundant in closed repeats.

2.3 Data structure

Biocurators annotate structures using some online available tools. The Biocomputing UP lab uses a software they developed and called the REFRACT annotation tool `https://tool.repeatsdb.org/load/`. This software allows biocurators to define regions and units boundaries and to assign to each region a specific annotations based on the classification described in table 1.6.3. Annotations can then be downloaded as json files which are the starting point for the definition of the dataset. Indeed after reading the json files using the Pandas Python Library i obtained a dataframe having a structure similar to the one in 2.1 where only most important columns are highlighted: each row corresponds to a single unit. where:

- start and end refer to the unit boundaries following the residue id numbering defined in the PDB files
- repeatsdb_id corresponds to the specific structure (PDB) and the chain in which that units was found. The first 4 letters represent the structure, while the fifth letter is the

start	end	repeatsdb_id	class	topology	fold	clan	UniProt_id
442	478	4zgcA	Closed repeat	Propeller	Five and six blade propeller	Kelch domain	Q8IDQ72
479	478	4zgcA	Closed repeat	Propeller	Five and six blade propeller	Kelch domain	Q8IDQ72

Table 2.1: Dataset structure

chain

- Class, topology, fold and clan are the annotation assigned to the specific region in which that unit was found
- UniProt_id is the id used in UniProt to identify the protein.

Even if it is not present in the table, a specific unique identifier has been associated to each unit too.

2.4 Data filtering

The SRUL cannot be made of the same amount of data in my dataset (17315 units). Indeed we want our dataset to be as small as possible but at the same time we want to maintain the largest possible amount of variability in order to be able to build a library of generic representatives. Based on the fact that similar sequences have similar structure [15], I reduced the structural redundancy by looking at the sequence similarity, therefore removing PDB structures that have the same sequence. PDB structures are generally fragments of the full protein. In order to remove redundant structures I have downloaded from `https://www.uniprot.org/` the UniRef90 ids associated to all UniProt protein in my dataset. The UniRef90 provide clustered sets of sequences from the UniProt Knowledgebase identified by a commond Id to all proteins sharing at least 90 % of sequence similarity and a 80 % overlap with the cluster representative, the so called seed. At this point i grouped the available units by their UniRef90 associated to the PDB chain to which they refer. As a consequence units of PDBs associated to nearly identical protein sequences were in the same groups. Then i went through each group and I selected one representative for each clan and each unit length. By using this procedure i was able to drastically reduce the dimension of my dataset: from 17315 to 6372 units as shown in table 2.2 the number of PDBs involved almost halved, since most of them are redundant be-

	units	pdb chains	UniProts
before filtering	17352	2119	917
after filtering	6372	1212	914

Table 2.2: Statistics about the dataset before and after filtering

cause they are associated to the same sequence. On the other hand, only three UniProts were discarded, since they were falling on the same UniRef90 cluster, thus we were able to maintain all UniProts with at least one PDB associated. In figure 2.3 is shown the distribution of units topologies after filtering we can notice that all of them are still present. The most frequent topologies got a more drastic decrease while the least present got a more restrained cut except for beta-trefoil and alpha/beta trefoil. Alpha-solenoids, propellers and TIM-barrels are still prevalent but they were extremely reduced in respect to other topologies. Before generating

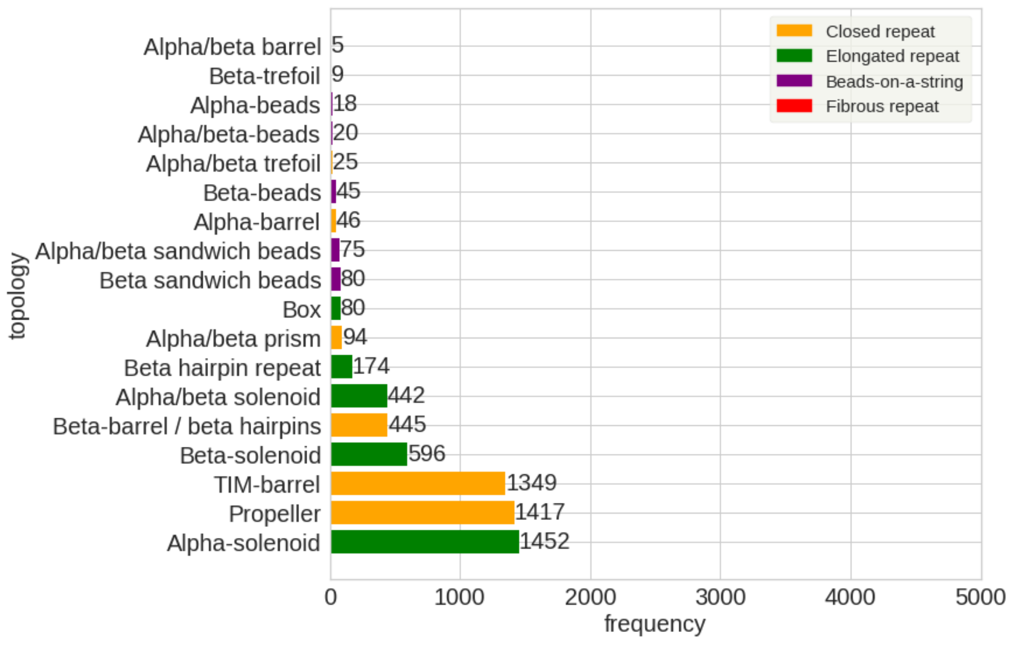

Figure 2.3: Unit distribution across structural topologies after filtering.

the SRUL we are also interested to notice how the topologies differ in the length of the units of their region. In this regard it could be interesting to plot what is the average unit length in each topology and how much can vary as shown in figure 2.4 and 2.5.Closed repeat on average have a higher average unit length in respect to alpha-solenoid but lower than beads-on-a-string. The average unit length is generally between 40 and 50 for closed repeat, except for TIM-barrel and beta-barrel/beta-hairpins having an average unit length under 40. The standard deviation looks much higher for propeller and beta-barrel/ beta hairpins while alpha-beta barrels looks more homogeneous. Elongated repeats are more variable: Box repeats are completely outliers for elongated repeats and they have the highest average unit length, followed by solenoid but with a difference of more than 20. Beta-hairpins, alpha/beta solenoid and beta-solenoid, on the other hand, are closed to each other between 20 and 25. The standard deviation is stable between 7 and 8 across all topologies except for alpha-solenoid where it reaches a value close to 20. This could be also due to the high prevalence of alpha-solenoid in contrast to the other topologies. Looking at beads-on-a-string we can notice how there are not topologies having an average unit length lower than 55. Beta-sandwich and alpha/beta sandwich beads seem the regions having the longest units. The standard deviation ranges between 20 and 40 with a peak of 36 for beta-beads and it is extremely higher than the one found in other classes certainly due to fact that beads-on-a-string have the longest units.

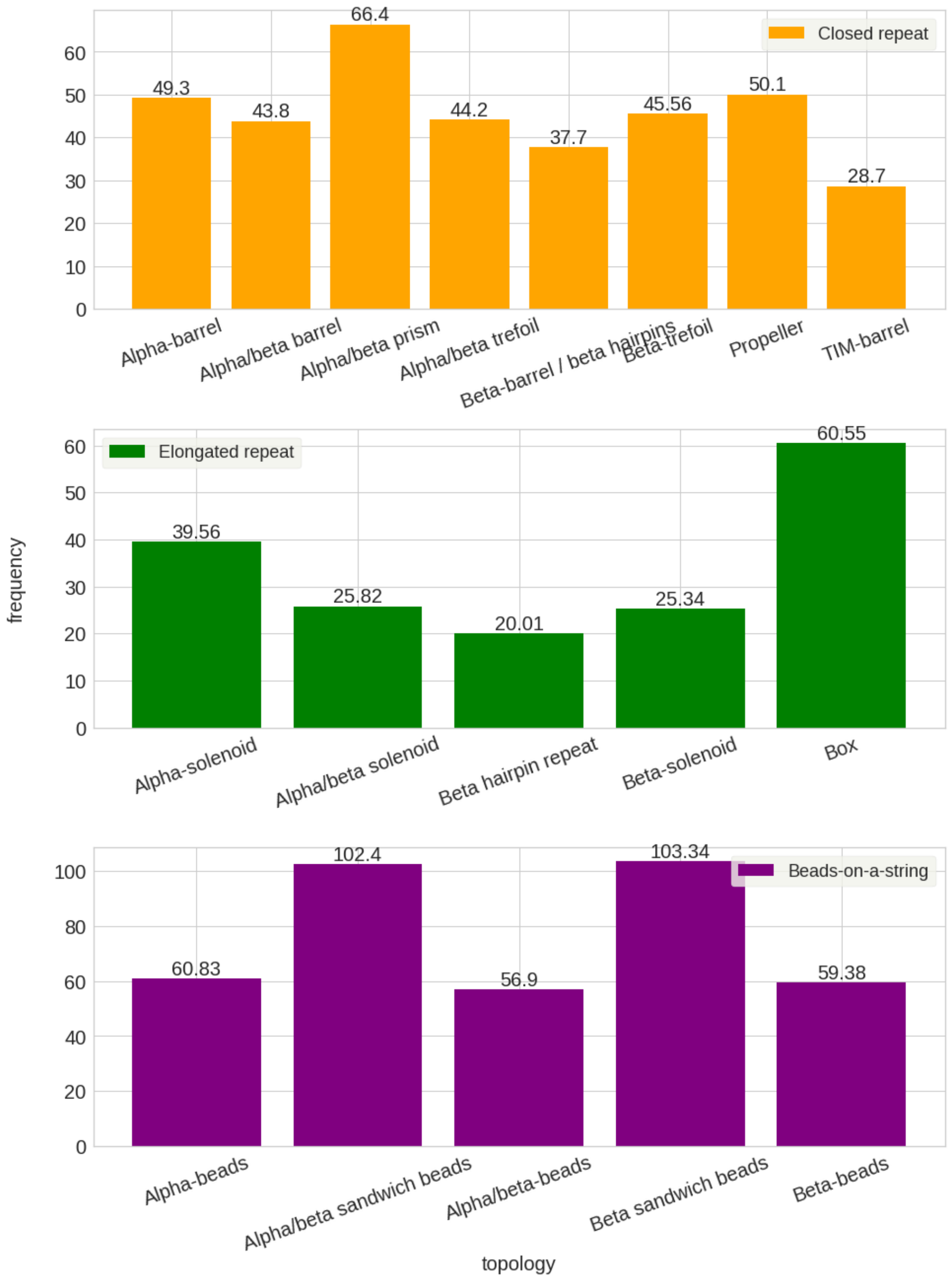

Figure 2.4: Barplot showing the average unit length for each topology

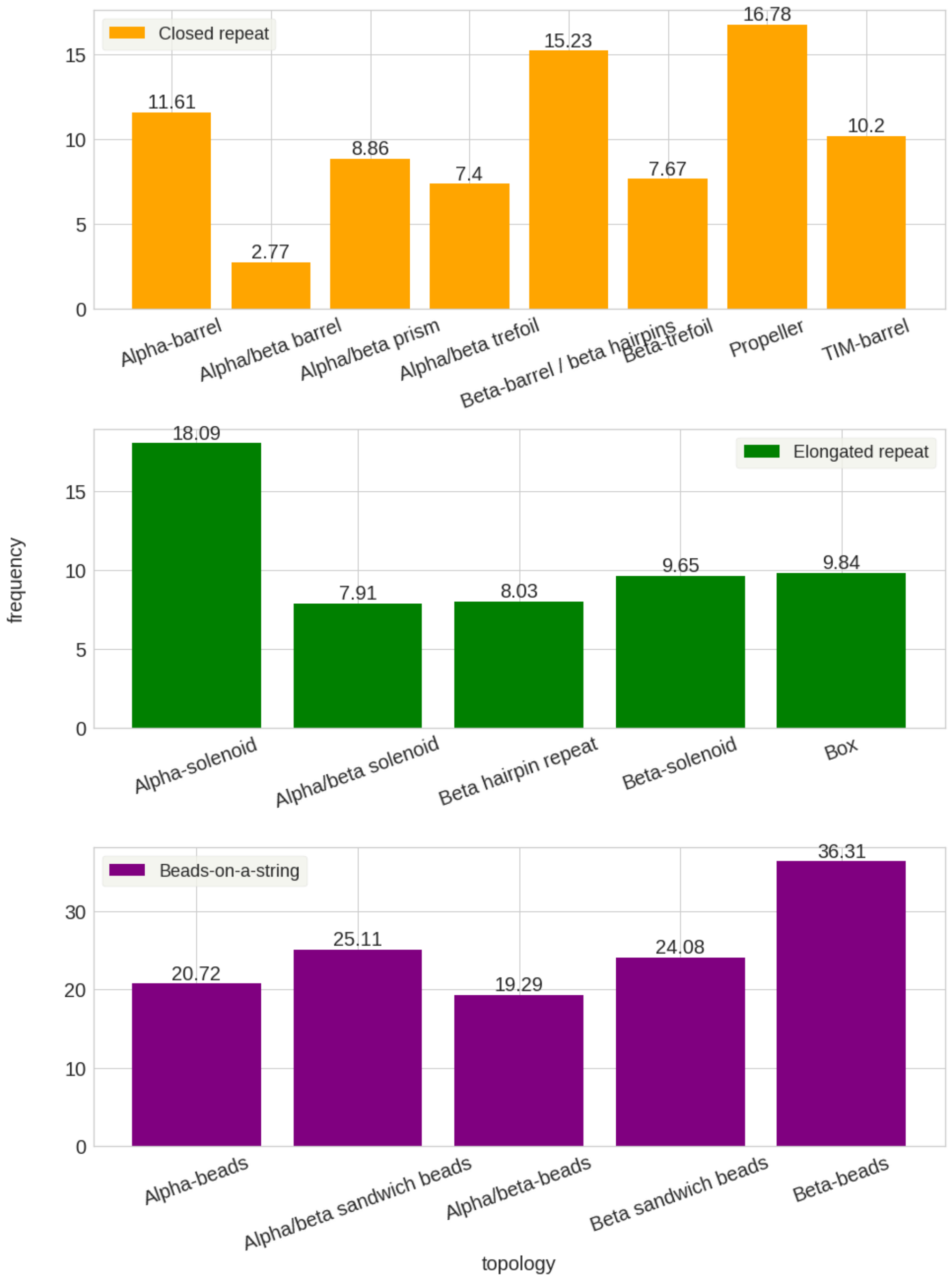

Figure 2.5: Barplot showing the standard deviation for units lengths for each topology

2.5 Data storage

The generation of the SRUL exploits the information contained in the filtered dataset organized as shown in table 2.1 where the start and end for each stored unit and the PDB chain in RepeatsDB are specified. The latter was used to download from the PDB Protein Data Bank all the structures associated to all units. The former, instead, was necessary when by iterating on the dataset i selected from the PDB files the three-dimensional coordinates of the α-carbon associated to each unit that i saved in a csv files along with the annotations contained in the original dataset. For each unit an incremental unit identifier was also assigned. The final SRUL structure is shown in figure 2.3.

residue	**unit id**	**atom**	**X**	**Y**	**Z**	**class**	**topology**	**fold**	**clan**	**PDB**
PHE	0	CA	-17.175	-14.43	-2.68	..	..	..	..	4zgcA
PRO	0	CA	-16.705	-16.44	0.552	..	..	..	..	4zgcA
LEU	0	CA	-13.299	-17.122	2.071	..	..	..	..	4zgcA

Table 2.3: SRUL library structure

3 SRUL clustering

The aim of this chapter is to explain why clustering was performed and how an optimal number of clusters was empirically chosen. First i will calculate three different matrices for pairwise all-versus-all comparison between units and i will produce some advanced scatter and density plots to highlight differences and similarities among classes and topologies. The second part of the chapter is focussed in the pre-processing steps required before applying clustering: normalization and dimensionality reduction. Normalization is required to perform dimensionality reduction which has been chosen to deal with the high dimensionality of the dataset. After that the hierarchical clustering approach is explained and the empirical procedure used to select the optimal number of clusters is presented.

3.1 Unit distance

After generating the SRUL, another important step consists in doing a more in depth analysis between topologies and classes where we want to highlight similarities and difference between units by taking advantage of the three-dimensional coordinates of the α-carbon we have retrieved from PDB files for each unit of the SRUL. In order to gather these kind of information a matrix representing the structural similarity between units pairs can be useful. It is calculated by performing a pairwise all-versus-all comparison between unit. In this case i decided to build three matrices: one for RMSD comparison, on for TM-score and another for absolute unit length differences.

3.2 Superposition and sliding windows

The main problem when comparing units is that we should rely on alignments. Structural alignments algorithms, first, compute the alignments between two structures of variable length to identify similar regions and after that a superposition is made. If we want to use a quicker and more immediate approach we can use sliding windows. Sliding windows is an approach used in many algorithms when dealing with objects or data structures of different lengths. The technique works as it sounds: a window of a certain length is created over some data coming from a data structure. At each iteration this window will move by n positions defined by the users and capture different portions of the object. In our situation it is very useful because we are dealing with protein structures of different length where no superposition is possible. With the sliding window approach we can select the smaller structure which will become our window (having a window size W equal to the selected structure length) and superimpose our window with the first W residues on the target structure. At the next iteration we will superimpose our window with the residues on the target structure obtained by moving our window of n steps. For example, if n=1 we will move our window of just of one step and we will compare it with residues going from the second residue to residue W+1. An example of sliding windows is shown in 3.1 where two small sequences are compared. In our case we are dealing with structures, so instead of making calculations between aminoacids letter we will consider the three-dimensional coordinates related to the α- carbons of each residue. Furthermore since this approach will be applied to all structures of the dataset there will be occasions in which the length of the window is higher than the length of the structure on which we want to slide on. In this cases a unique superposition will be made just between the minimum length common set of residues between the two structures starting from the first, without making any further sliding window.

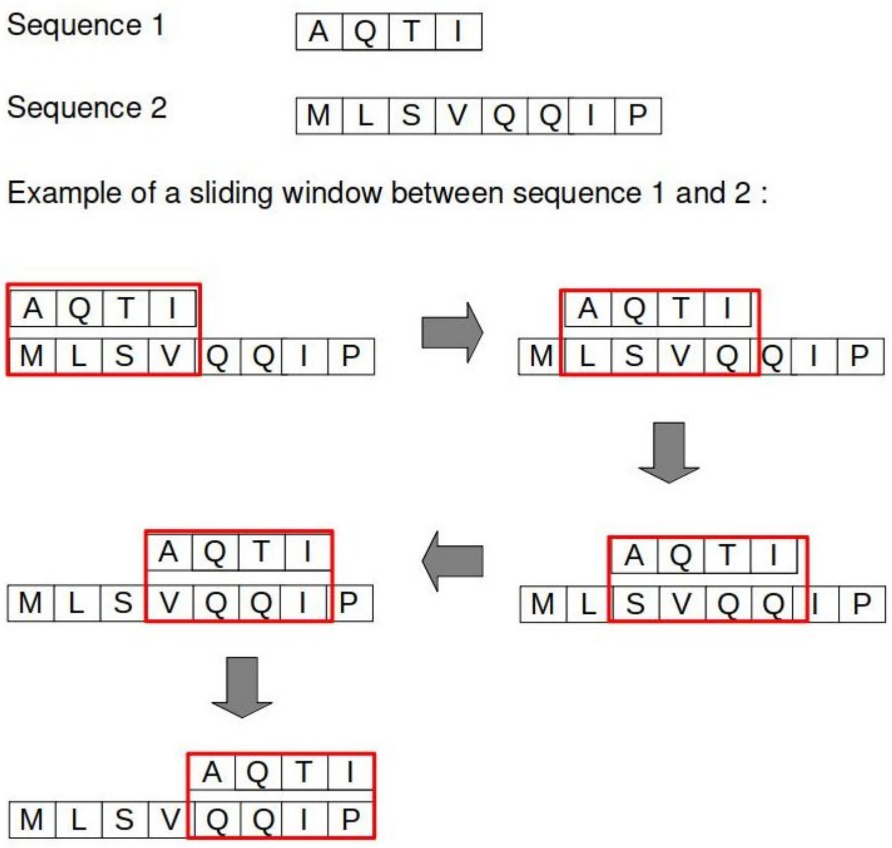

Figure 3.1: Sliding window between two sequences. The first sequence having a length of 4 slides over the second sequence of one step at each iteration. The red rectangle highlights the two subsequences on which the superposition is performed

3.3 Distance matrices

As highlighted in 3.1 three matrices for units comparison are generated. The minimum dimension of those matrices is 6372x6372 since we want to compare each unit with each other unit. However we will need at least another column to identify what unit is taken in consideration in each row. Moreover, just for convenience another column containing the PDB chain to which the unit refers is added. In the end the three matrices will have the following structure (in the figure matrix of RMSD is shown as an example): Summarizing: three different 6372x6374 matrices will be generated using an approach based on sliding windows in order to be able to superimpose structures having different length. It is important to notice that an approach

pdb chain	**unit id**	**unit 0**	**unit 1**	**unit 2**	......	**unit 6371**
6kivR	0	1.8	2.3	3.3	......	12.2
6kivR	1	3.2	4.8	3.3	......	12.2
⋮	⋮	⋮	⋮	⋮	......	⋮
5hhgE	6371	15.1	9.6	8.6	......	4.6

Table 3.1: Matrix of comparison of Root mean square deviation

based on sliding windows can produce an amount of fragments very variable depending on the difference in length between the two units. Anyway only the results related to fragment for which the TM-score is maximized and the RMSD minimized will be stored in the matrices.

3.4 Unit relationships

The units analysis has been carried out using the three matrices presented in the previous sections. Being able to determine the composition of your dataset, the difference in distribution among units belonging to different classes or topologies is essential to underline similarities and differences. Furthermore, the algorithm that will be presented in the next chapter is based on clustering of units contained in the SRUL. From this analysis we will be eventually able to understand if units of the same class could be grouped in different clusters but at the same time together with units of other classes. The first 4 plots represent the density distribution for minimum, median RMSD and maximum, median TM-score using the kernel density estimate to highlight the probability distribution of the variable under consideration in a smooth and continuous way. Those plots can provide more information than histograms because they do not depend on the number of bins chosen.

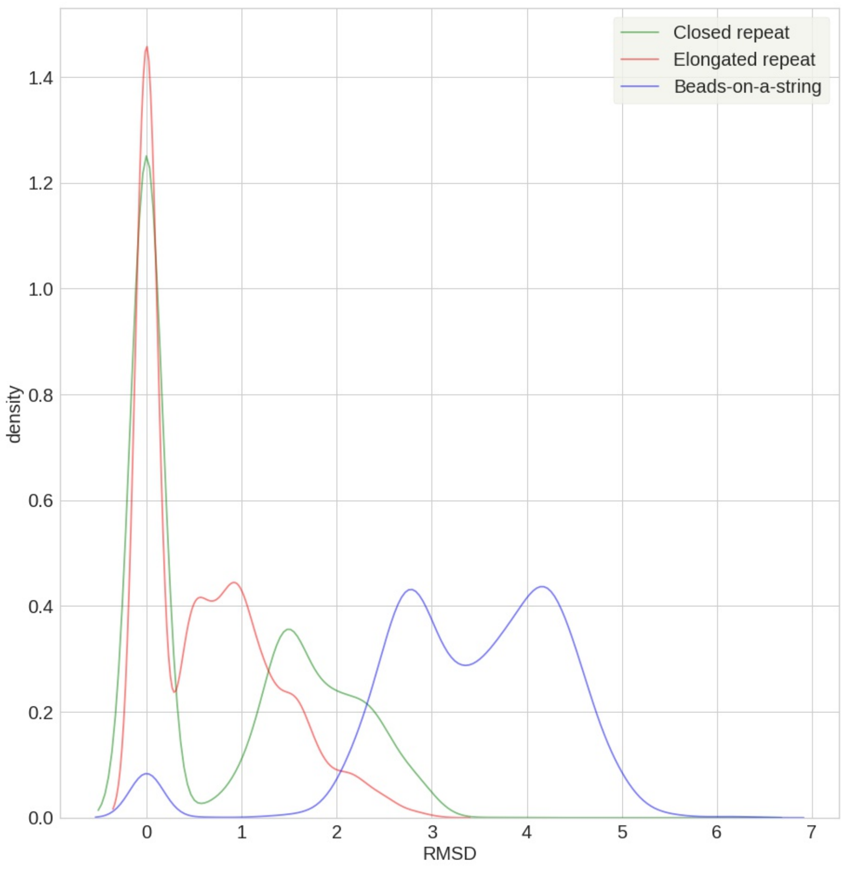

Figure 3.2: density plot for minimum RMSD

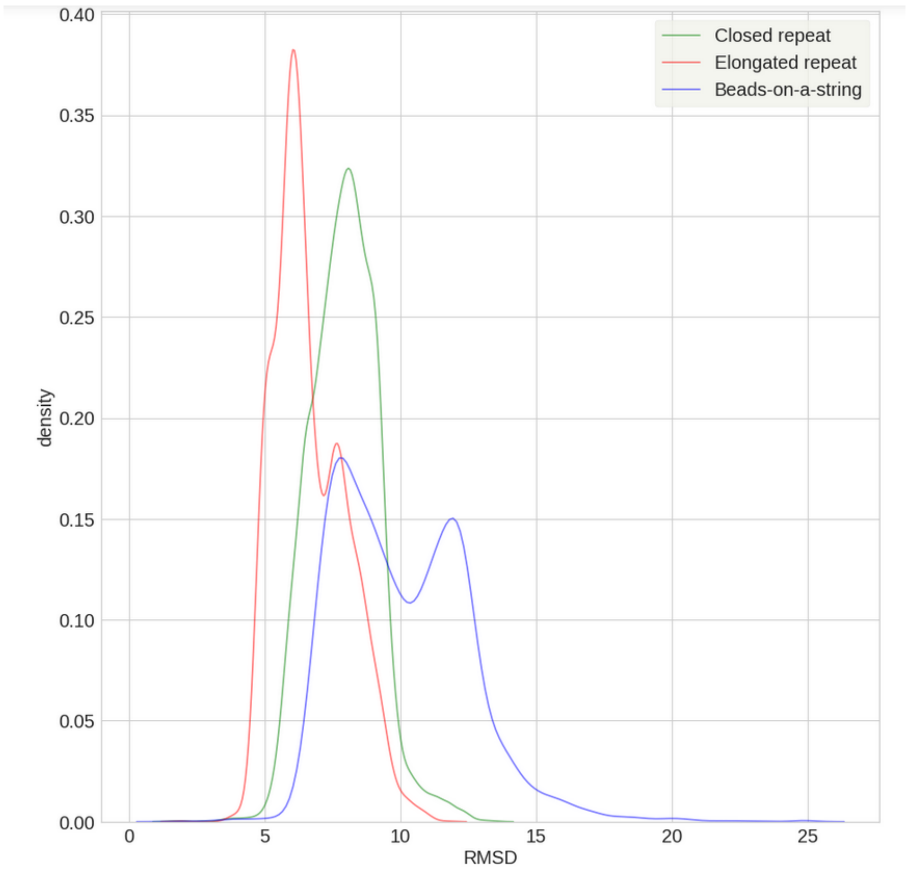

Figure 3.3: density plot for median RMSD

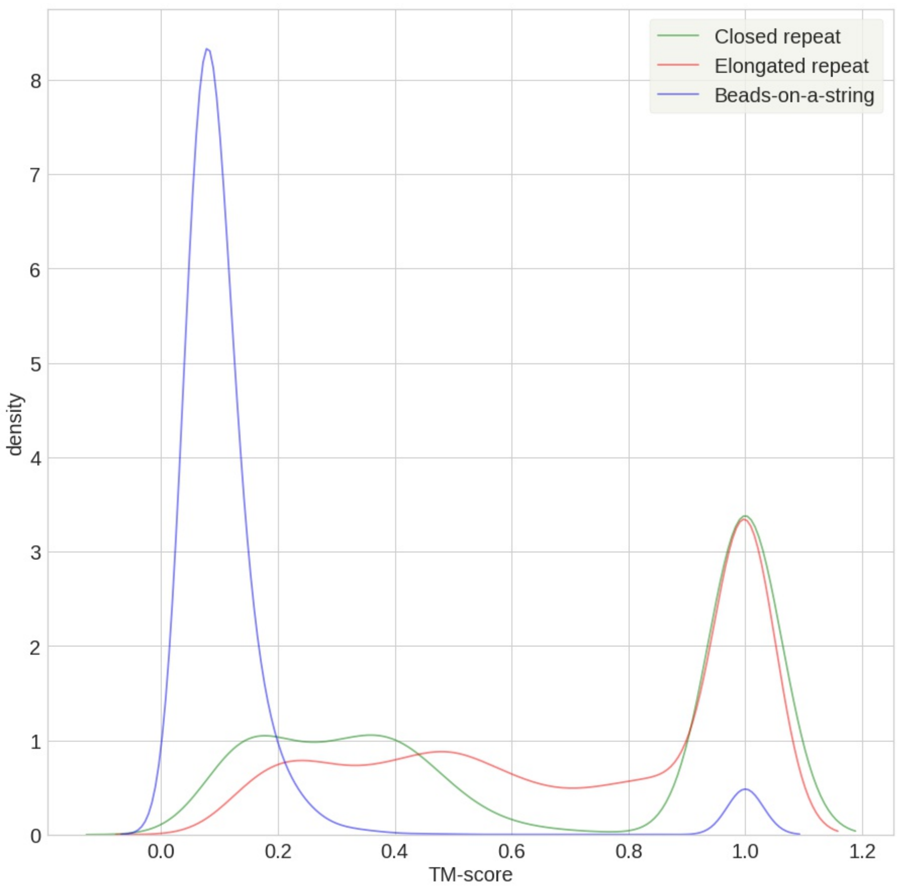

Figure 3.4: density plot for maximum TM-score

Figure 3.2 represents the minimum distribution for the three classes. The minimum has been computed using the RMSD matrix and by retrieving the minimum value from each column. The graphs shows three multimodal distribution on for each class: The maximum density is in 0 for elongated repeats and closed repeats even if elongated repeats peak higher in density. Other two near peaks can be found between 1 and 2 even if the distribution of closed repeats is slightly shifted towards the right. Anyway we can say that elongated and closed repeats have a similar distribution and very shifted to left which is the sign that they share many similarities with other classes. On the other hand beads-on-a-string have a completely divergent behaviour: the peak in 0 is very small and most of the units have a minimum RMSD between 2 and 5 symptom of a lower degree of similarity with units belonging to other classes compared to closed and elongated repeats. The reason of this behaviour is dictated probably by the average length of units which is often double than the average length of elongated and

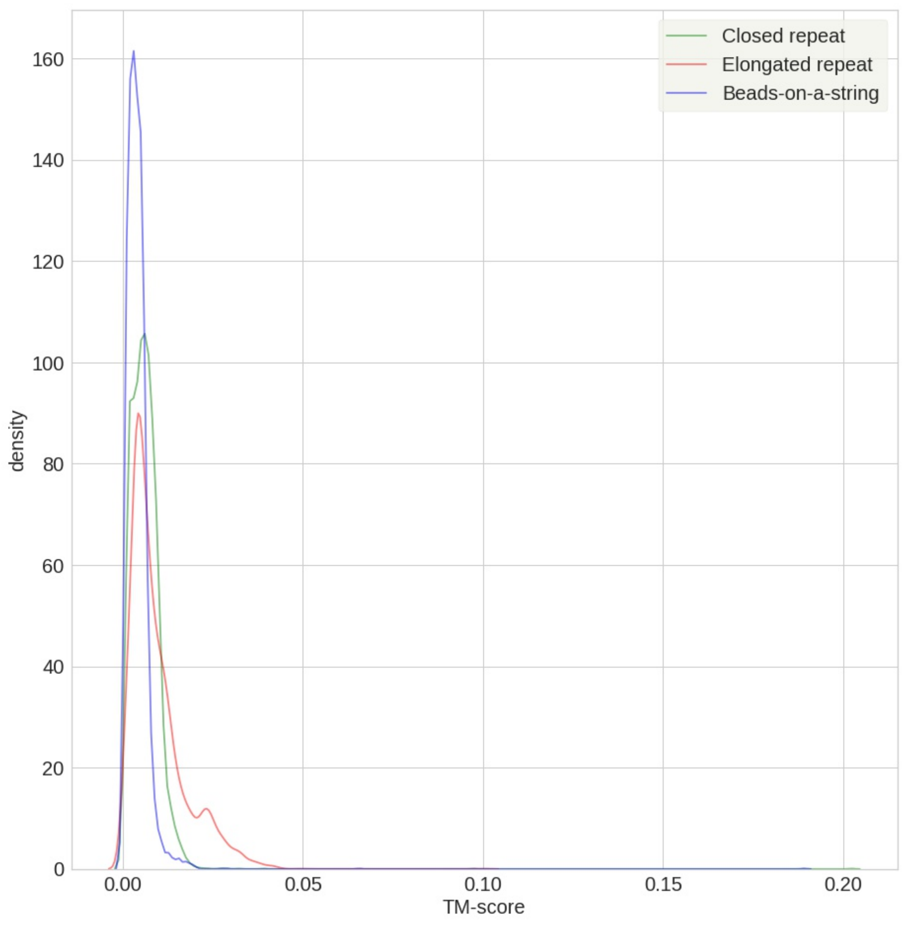

Figure 3.5: density plot for median TM-score

closed repeats. The plot in 3.3, shows, instead the distribution of median RMSD calculated by computed the median RMSD over each column of the matrix. Again, this plot appears to confirm what shown in 3.2: elongated repeats is the class having most peaks shifted to the left, thus they share more similarities with units belonging to other classes, closed repeats behave in a similar way and beads on a string, instead, are the most shifted towards the right. However, overall differences are less visible and marked. 3.4 and 3.5 represent the same plots but considering TM-score instead of RMSD. We expect those plots to be smoother than the previous one because TM-score is less sensitive to small deviations. Neverthless, the density plot for maximum TM-score 3.4 still captures differences very well. The behaviour is, as expected, exactly the opposite in respect of what highlighted in RMSD plots: elongated and closed have a nearly identical trend, while beads-on-a-string have a huge peak at 0.1 TM-score recalling again their low similarity compared to units belonging to the other two classes. On the other hand, figure 3.5 depicts a different but confirmatory scenario: the three curves are almost overlapping and peaking at a very small TM-score value. Again, even if they contain the least amount of units, beads on a string are the ones peaking the highest at very low TM-score values confirming they contain structurally unique units.

Another interesting plot is 3.6 where TM-score standard deviation has been computed for each column of the matrix and represented separately for the three classes. Closed repeats and beads on a string have a less variable standard deviation and more clustered near 0.01. Elongated repeats, instead, have more sparse points in a range between 0.01 and 0.04 highlighting the increased amount of variability characterizing this class.

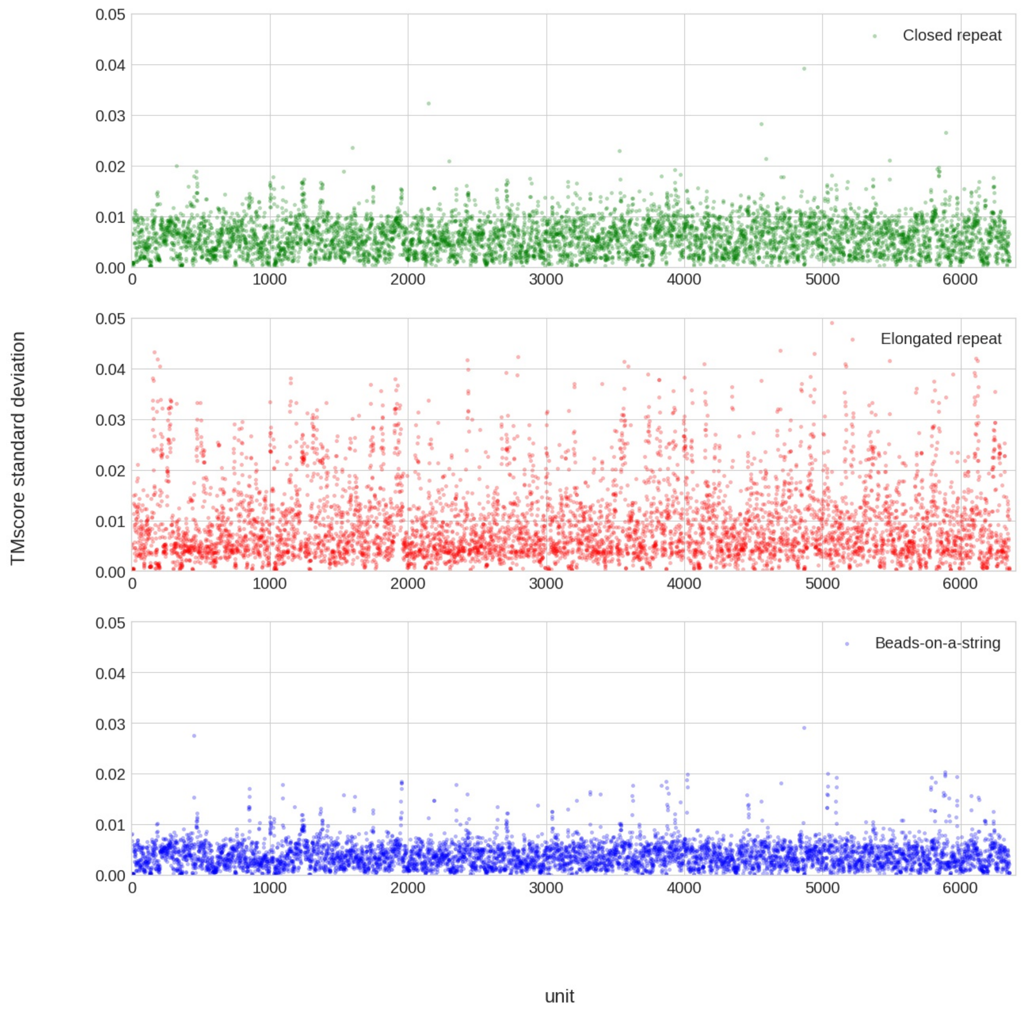

Figure 3.6: scatterplot showing the TM-score standard deviation for the three classes

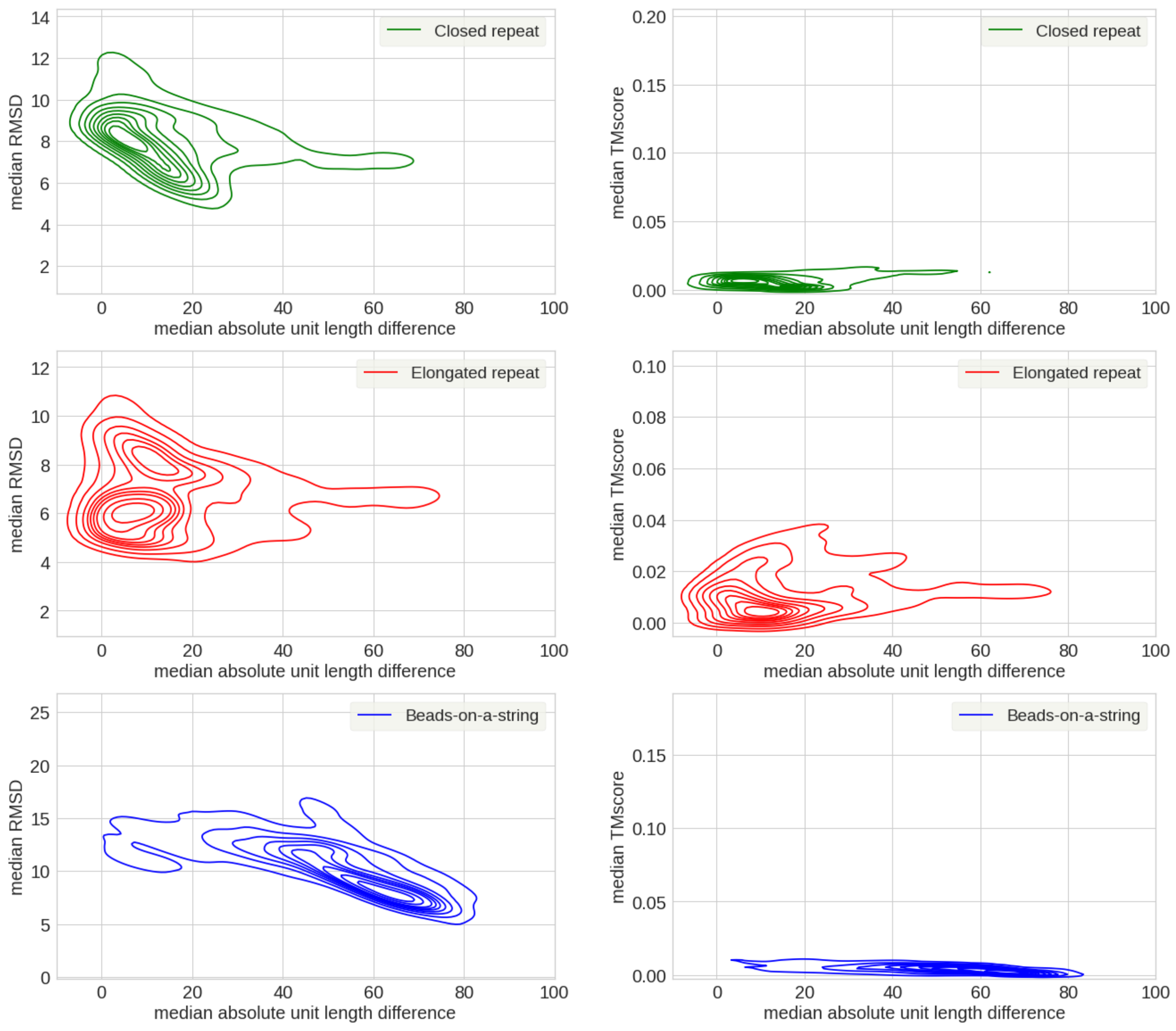

Figure 3.7: density plots comparing median RMSD vs median absolute unit length difference and median TM-score vs median absolute unit length difference

Until now we have observed the distribution of RMSD and TM-score without taking into consideration also the median absolute difference in unit length. Figure 3.7 represent bivariate probability distributions between RMSD and TM-score against median absolute unit length divided per class. On the left average unit RMSD is compared with median absolute unit length difference: for closed and elongated repeats most of lowest RMSD and highest TM-score matches are present when the average unit length is small. On the contrary beads on a string show the opposite behaviour: RMSD and TM-score are lower when the median absolute unit length different is higher. This is probably caused by the fact that is very difficult to find a good match between beads on a string units and other class units, but at the same time some of the structural elements contained in the long beads on a strings units share a high similarity with units that are part of the other two classes. Finally it is fundamental to notice that we want to observe general trend, hence we are considering medians computed on highly populated classes containing lots of structurally different topologies so it shouldn't be surprising that in all the cited plots RMSD values are not that low and TM-score ones are not that high. Indeed median was chosen over mean because it is less sensitive to outliers. Going deeper in the RepeatsDB classification a better analysis can be done by looking at topologies. As i did for classes i plotted the bivariate probability distributions average absolute unit length difference against RMSD for topologies. In figure 3.8 i plotted closed repeats: the distribution as expected is more shifted to the left and it id more similar between propeller, beta-trefoil, alpha/beta trefoil, alpha/beta barrels and alpha beta prism having a vertically elongated distribution. Beta-barrel and TIM-barrel are instead more flat, probably also related to the high variability in unit lengths and their average unit length which is lower than other topologies and which comes clear from barplot presented previously in 2.4 and 2.5. The distribution of elongated repeats, shown in figure 3.9 is completely different from the one of closed repeats and spread horizontally. Except for box repeats there are always a few units showing some similarities with other ones even when the difference in length is high. Alpha-solenoids, for example, could share a good structural similarity with the alpha-helices contained in the alpha-beads regions of beads-of-a-string. The very short beta-hairpins repeats could match with beta-sandwich beads too. Beads-on-a-string represented in figure 3.10 share all similar distributions but shifted in very different ways. Beta-sandwich beads and alpha/beta beads share more similarities when the average difference in length is high, because they are structurally more close to motifs of units of other classes. Alpha/beta-beads, beta-beads and alpha-beads, instead, have a distribution very shifted to the left and clustered where the average absolute unit length difference is between 25 and 40.

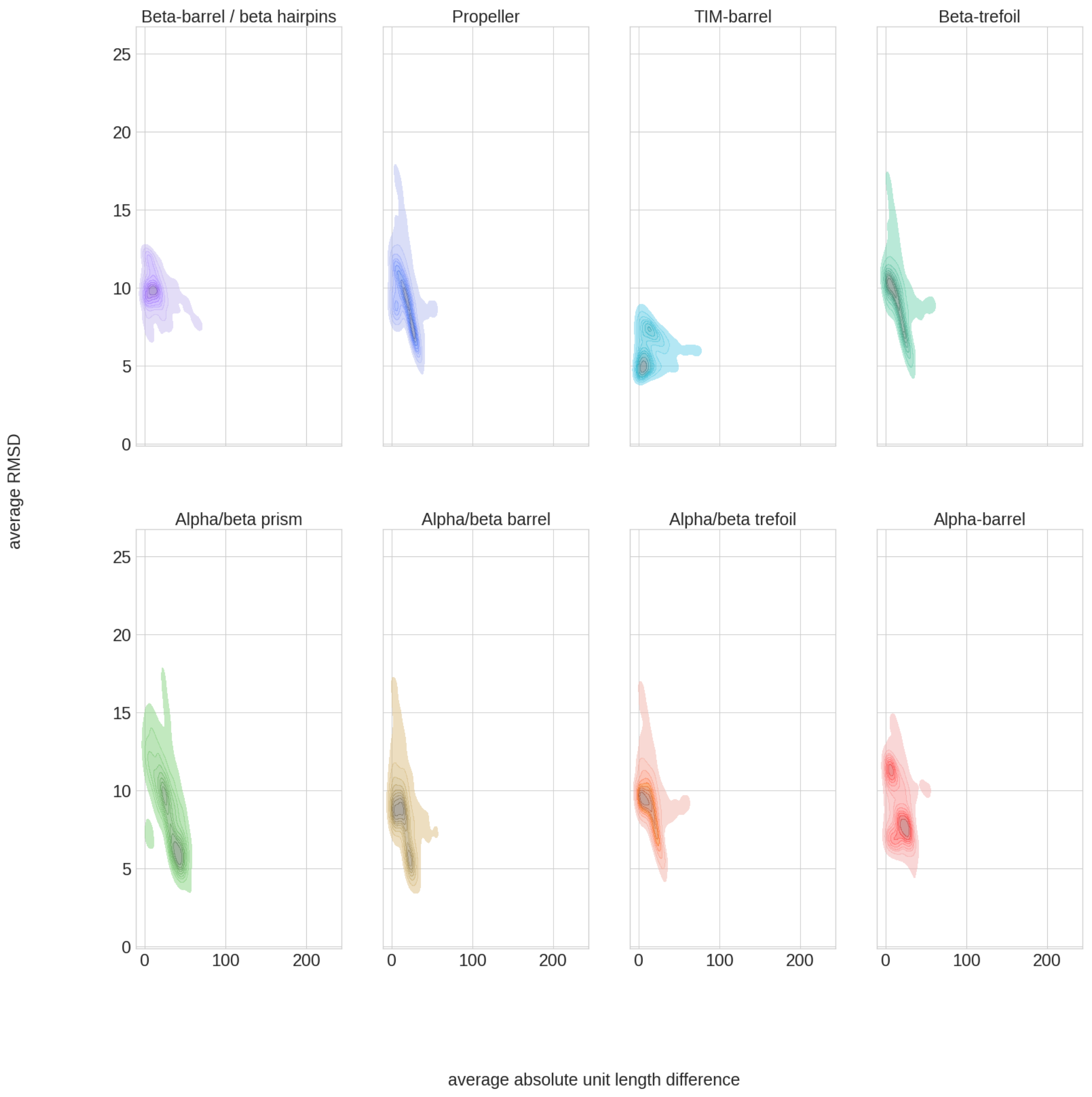

Figure 3.8: density plot comparing average rmsd with average absolute unit length difference for closed repeats

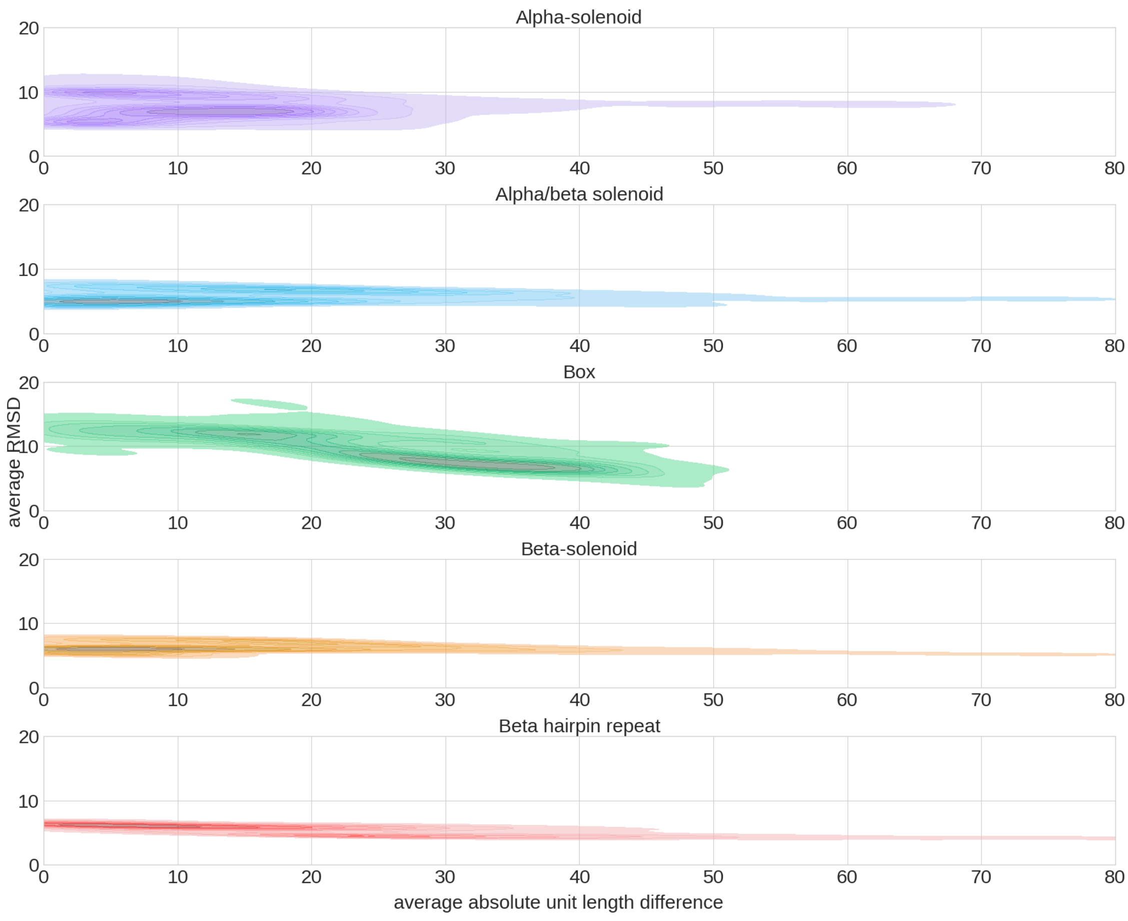

Figure 3.9: density plot comparing average rmsd with average absolute unit length difference for elongated repeats

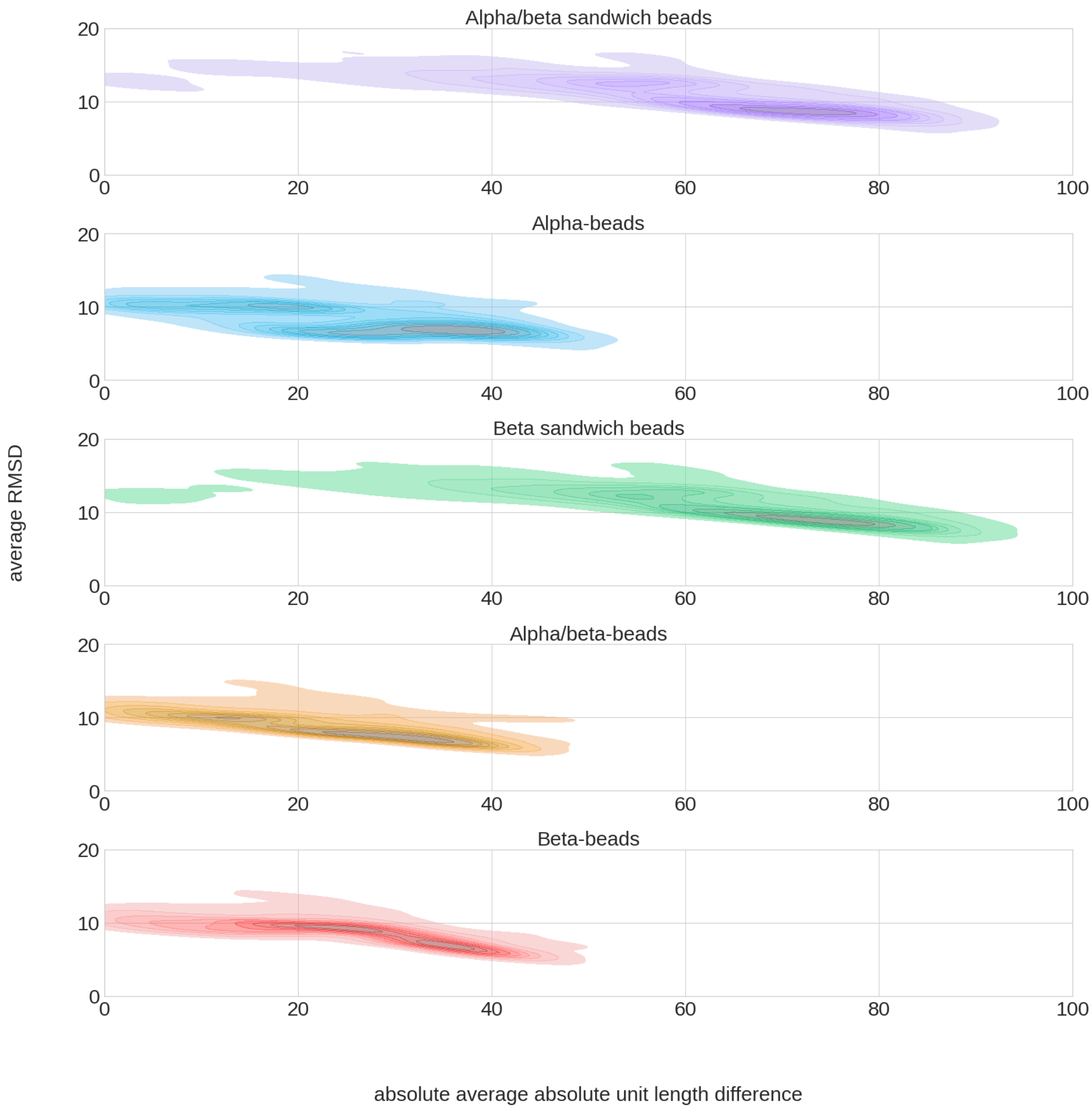

Figure 3.10: density plot comparing average rmsd with average absolute unit length difference for beads on a string

3.5 Preparing data for clustering

As mentioned in the previous section the final purpose of comparison matrices is to use them to generate a dataset where each row represents a units and each column should contain the RMSD or the TM-score derived by the comparison with any other unit. The dataset should then be used to apply a clustering algorithm to make the search in the SRUL by the algoritm, which will be presented in 4, faster and more targeted as possible. For this reason for the final dataset i just decided to use the matrix of RMSD and not the TM-score because it is more sensitive to small deviations. A t the same time, however, we want to provide information also about the difference in unit length to group in the same cluster units with a similar length. Speaking of which, i decided to merge the RMSD matrix and the matrix of lengths in a unique matrix with the same number of rows and double the number of columns. As before the first two column of the matrix will contain the unit id and the pdb chain to which the unit refers too. The columns from 2 to 6374 of each row will contain the RMSD values, while the one from 6374 to 12746 will contain the absolute difference in length. In the end our matrix will be 6372x12746 with a number of columns extremely higher than the number of rows. A dataset of this kind is called a high dimensional data dataset because the number of features is higher than the number of observation. In this scenario a phenomena called the curse of dimensionality (term coined by Bellman [16]) occurs: objects become very difficult to interpret and visualize and data become very sparse meaning that the amount of data necessary to balance the high number of features is exponential, thus if you don't have many observation data will become sparse in the space and most of the methods coming from machine learning and data mining wont work anymore. To overcome this problem first i normalized the values in the dataset and then i applied a dimensionality reduction technique called Principal component analysis.

3.6 Normalization and dimensionality reduction

In order to normalize the dataset the MinMax scaler from sklearn* has been used to to scale each feature individually such that the minimum value is 0 and the maximum one is 1. The reason behind normalization is that PCA, which is a variance maximization algorithm, require data to be defined in the same range, otherwise it will project data in a wrong way. More in general Principal component analysis is used during data processing and before submitting data to a predictive model. It is a dimensionality reduction technique which projects each observation

*https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html only onto the first few principal components to obtain lower dimensional data while preserving most of the available variance. As a consequence the first components will be the ones explaining the highest amount of variance. It can be proved, using some linear algebra, that the principal components are the eigenvectors of the data covariance matrix that can be easily calculated using the singular value decomposition (SVD). In order to choose a good number of components i first computed the amount of variance explained by each single components obtained by the features of our dataset: Figure 3.11 shows the amount of variance explained by the first 13 components. It can be noticed how with only 13 features we can explain 0.96 of the variance of the entire dataset which is actually a lot. The remaining 12731 account for only 4 % of the variance so they can be easily discarded. Summarizing by using PCA we were able to decrease the dimensionality of our dataset from 12744 numerical features to only 13 solving the issue of curse of dimensionality.

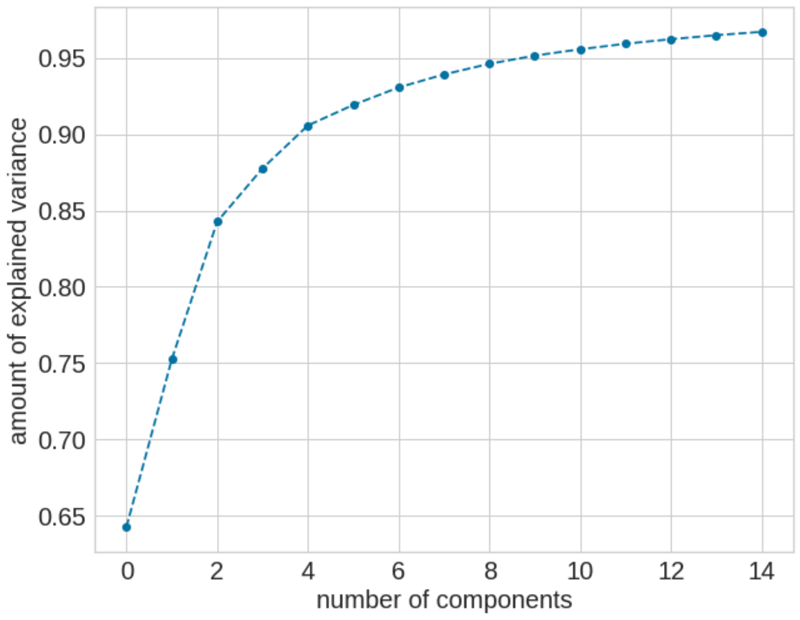

Figure 3.11: Amount of explained variance by the first 13 components

3.7 Clustering

Our final dataset is made of 6372 units and 13 features. Our algorithm, which will be further explained in 4.1, predicts units boundaries by doing multiple comparisons of the units contained in our SRUL and the input structure. For this reason it is mandatory to design a strategy to select the best candidates that can have some structural similarity with our input and to discard instead units that could not provide any information and make the algorithm slower. In this regard an optimal technique coming from unsupervised learning and called clustering can be very useful. Clustering is the procedure of grouping objects sharing a strong similarity: the idea behind this procedure establishes that observations in the same group should be more similar than those that are located in other groups. Clustering can be performed in different ways: some algorithms like the gaussian mixture use statistical distributions, some others like DBScan are density based models which try to connect dense regions located in the space. Our approach, instead, is based on hierarchical clustering which seeks to build a hierarchy of clusters by optimizing an objective function that defines the distance between observations. The graphical representation commonly used in hierarchical clustering is called dendrogram, a rooted tree representing how clusters are arranged. Different kind of objective functions exist and this is what makes hierarchical clustering so flexible and useful even when not enough information are available for the dataset under analysis. Each objective function have its strength and weaknesses and privilege clusters of a difference size. For our purposes we want clusters to have a small size (between 10 and 15 units) but with each clusters having units sharing the maximum possible similarity. For this reason among the linkage criteria available i decided to use one very common in bioinformatics called WPGMA. WPGMA is able to produce a dendrogram rooted tree from a similarity matrix: it is an agglomerative bottom up procedure very used in biology that recalls the behaviour of a molecular clock. A molecular clock is a procedure applied to biochemical data to estimate when two life molecules diverged in time. According to the molecular clock theory the rate of evolutionary change of any protein is almost constant over time and over different lineages. It is very common when comparing two proteins, DNA or RNA sequence. The WPGMA assumption for building the dendrogram is exactly the same of a molecular clock: the distance from the root to every branch of the tree is equal. Initially each unit is assigned to one single cluster as a singleton. At each iteration the two nearest cluster say i and j are combined in a unique cluster. The distance between the newly obtained higher level cluster and the other clusters is computed as the arithmetic mean of the average distances between members of clusters k and cluster i and cluster j and k (see formula in figure 3.12).

$$d(u,v) = \frac{dist(s,v) + dist(t,v)}{2}$$

Figure 3.12: update formula for the WPGMA algorithm

Cluster are made of lots of units, hence it is essential to find one unit for each cluster that can somehow summarize the main shared characteristics . For other clustering technique, like K-means it is way easier because a point called centroid is directly computed by the algorithm and can be used to instantly find cluster representatives by just calculating the distance between units and the respective centroids. With hierarchical complex things get trickier: in order to achieve this i first computed for each group the average cluster unit: the average cluster unit is a fake unit calculated by averaging cluster units based on its features. The average cluster unit is then used to find its nearest unit in the cluster which it will become the representative for that specific cluster.

3.8 Number of clusters

One of the most difficult step when dealing with cluster is to select a correct and functional number of clusters. There is no mathematical theory for unsupervised learning that can help on this and even heuristic methods commonly available (gap statistics, elbow methods, silhouette score) are more suited for centroid based clustering tecniques or are not tied for the specific task you are required to solve. Moreover elbow method is often inexact if clusters are very different size and gap statistics is not that easy to compute. For this reason i decided to prepare a set made of 15 structures to submit to RepeatsDB-Lite 2 (the new algorithm presented in 4.1) representing each one a different topology to verify empirically how the algorithm described in 4.1 performs in terms of accuracy, precision, recall varying the number of clusters (from 100 to 1000 using a stepsize of 10). Theoretically speaking, the number of cluster should be neither too big or too small. If it is too small clusters will become very populated, hence for most of the inputs the algorithm will go through many units in a single clusters. If the number of clusters is too high most of them will be singletons, thus the algorithm will have to go before many units before providing a valid solution. The graph in 3.13 highlights that the algorithm performs very well when the number of cluster is less than 100 or greater than 300. Infact between 100 and 200 there is a sharp drop in the overall performances.However the main issue with this graphs does not take into account that some of the regions associated to the PDBs

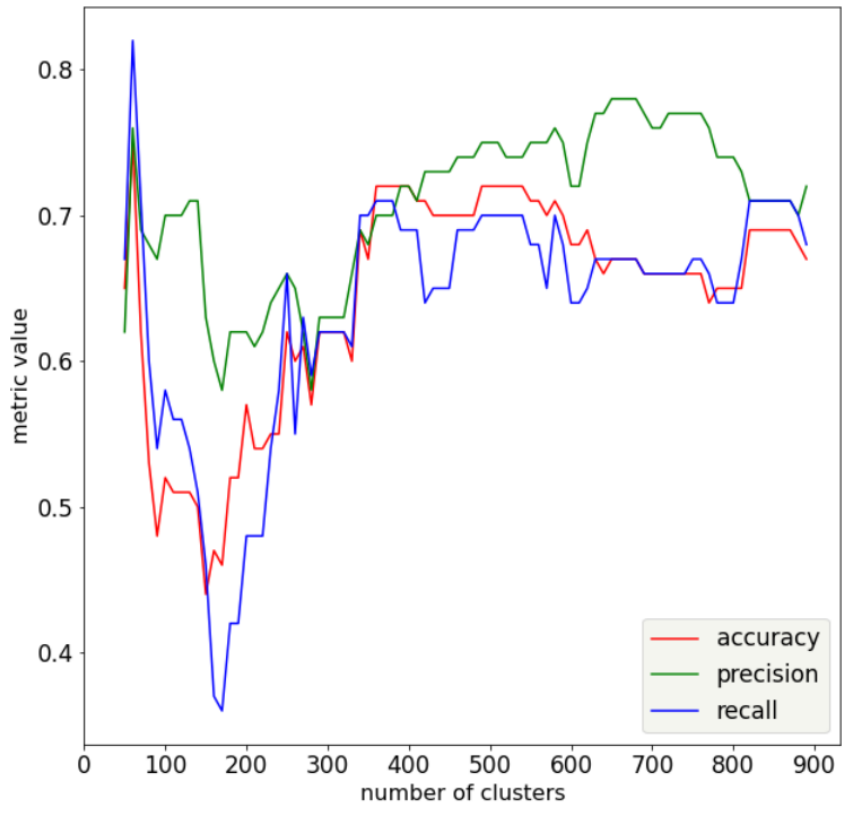

Figure 3.13: Performance of the algorithm (RepeatsDB-Lite 2) when varying the number of clusters

used for validation could be undetected. if we look at 3.14 we observe a step like decrease in the number of undetected regions: as we keep increasing the number of clusters the algorithm seems to be more precise in detecting all required regions. This is probably related to the strong inbalance of units in our SRUL: when the number of clusters is low it is more frequent that most of the cluster representatives belong to the topologies that are more present in the dataset, thus the algorithm fails to detect regions of the most marginal topologies. In the end 3.14 was useful because we could understand that even if precision, recall and accuracy have their highest peak when the number of clusters is less than 100, the number of undetected is high and never inferior to 4. For this reason the optimal number of cluster seems to be included between 500 and 600. Indeed 500 seemed to be the best number of clusters where a average unit precision of 75 % and a average unit recall of 70% was reached.

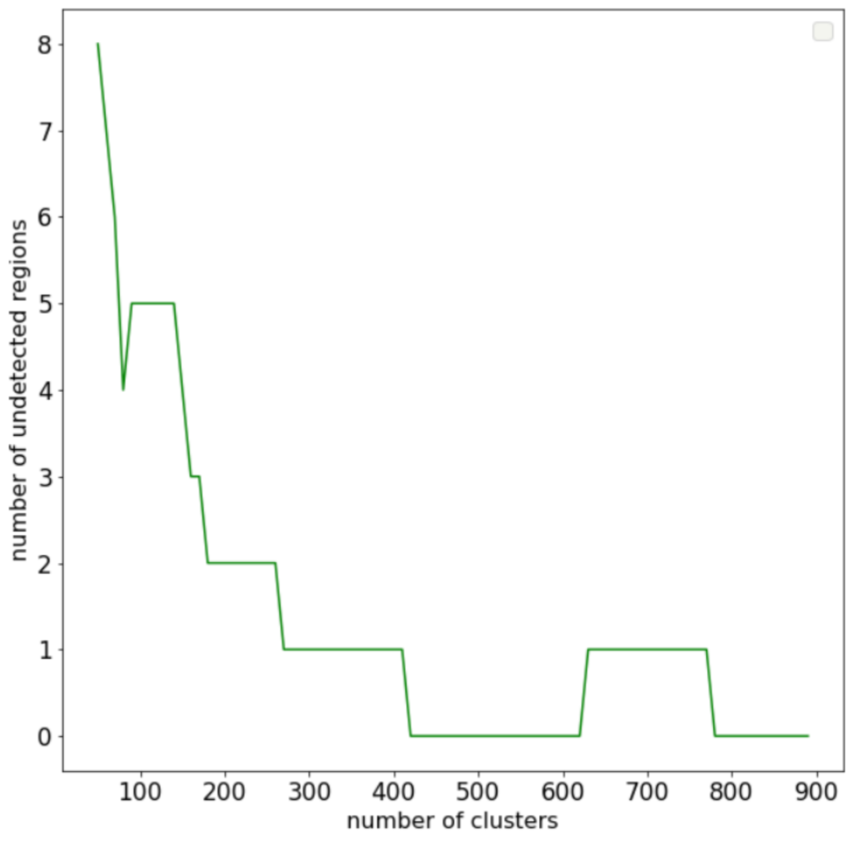

Figure 3.14: Number of undetected regions when varying the number of clusters

4
TRPs prediction

This chapter can be divided in two part: the first aims to present RepeatsDB-Lite 2, a new and faster approach based on superposition and relying on clustering to predict units and annotate regions extremely faster than RepeatsDB-Lite. Moreover, the technique used to validate predictions will be shown too. The second part of the chapter will show the results obtained by the new algorithm on a dataset coming from RepeatsDB. Therefore a comparison in terms of time and performances with RepeatsDBLite will be made. Finally, some real case use scenario will be presented highlighting how this algorithm could be useful to make prediction using automatically predicted structures from the primary sequence thanks to AlphaFold.

4.1 RepeatsDB-Lite 2 algorithm

The new algorithm, that from now on it will be called RepeatsDB-Lite 2, is shown in 4.1 and was designed to classify and predict repeated regions in tandem repeats proteins. It takes as input a single PDB chain or a list of PDB chain, the Structural Repeat Unit Library and a file containing for each unit of the SRUL the assigned label according to the clustering procedure explained in 3.7. First, if not present, the PDB file is downloaded from the PDB database [*] and then parsed by the algorithm while the SRUL is loaded in memory. Differently from RepeatsDB-Lite the PDB structure is compared only with some selected candidates of the

[*]https://www.rcsb.org/

SRUL to speed up the whole process: indeed, the input structure is compared with each cluster representative (computed as explained in 3.7) by making a superposition with sliding window of the unit with the input structure. This first superposition is essential because it can help to discard also very populated clusters that do not share any similarity with the input. After that the algorithm select the cluster for which the superposition with representative gave the highest TM-score and the lowest RMSD and compares the input structure with each unit of the cluster. Again, a superposition is made between the input structure and each component of the cluster but two times: the first time to select the three fragment of the input where the unit matches the best, the second time to re-superimpose the selected fragment which are unit candidate in turn with the input structure. The whole procedure is repeated for each unit of the cluster and the resulting fragments of the second superposition having a TM-score higher than 0.20 are maintained. It is worth to point out that only TM-score has been used here: infact it is less sensitive to small local deviations and it is normalized. While RMSD can go from 0 to infinity, TM-score is bounded between 0 and 1 so it's more prone to be used to set thresholds. After exploring all the cluster a first tentative of merging units with a strict filter is done following the procedure described in scheme 4.2: only the units having a TM-score higher than 0.30 and less than 5 gaps are taken into consideration. The one having the highest TM-score is the master unit and it is picked as first unit to build a path by minimizing gaps with preceding and succeeding units . If the result is not satisfying the algorithm picks another master unit by selecting the second with highest TM-score for a maximum of 10 times. If after trying many different master units no path is found hence three units were not paired together, filters are relaxed and the whole procedure is repeated for other 2 iterations by lowering the TM-score threshold. If, still no result is obtained the maximum number of gaps allowed is relaxed for a maximum of 2 times and the previously described iterations of TM-score is repeated(see table 4.1) . Moreover, If no result is found even when filters are fully relaxed the algorithm goes in the second best cluster to gather more potential unit candidates for a maximum of 5 times. On the other hand, when at least three units are found they become a candidate region and an annotation in terms of class, topology, fold and clan is inherited from the master unit and assigned. Moreover, gaps between units are labelled as insertions if greater than 7 or filled otherwise. After that the algorithm checks if the left fragment or the right one in respect to the found region of the input structure have more than 40 residues, and if it is that the case it cuts the structure and it repeats the whole procedure. When no more fragments can be processed the obtained candidate regions are compared: if candidate they share an identical topology annotation and they are found to be in proximity of each other they are merged together in a unique region. Results are then written in a csv file(Table 4.2). The whole algorithm has been written in Python using the library Biopython[†].

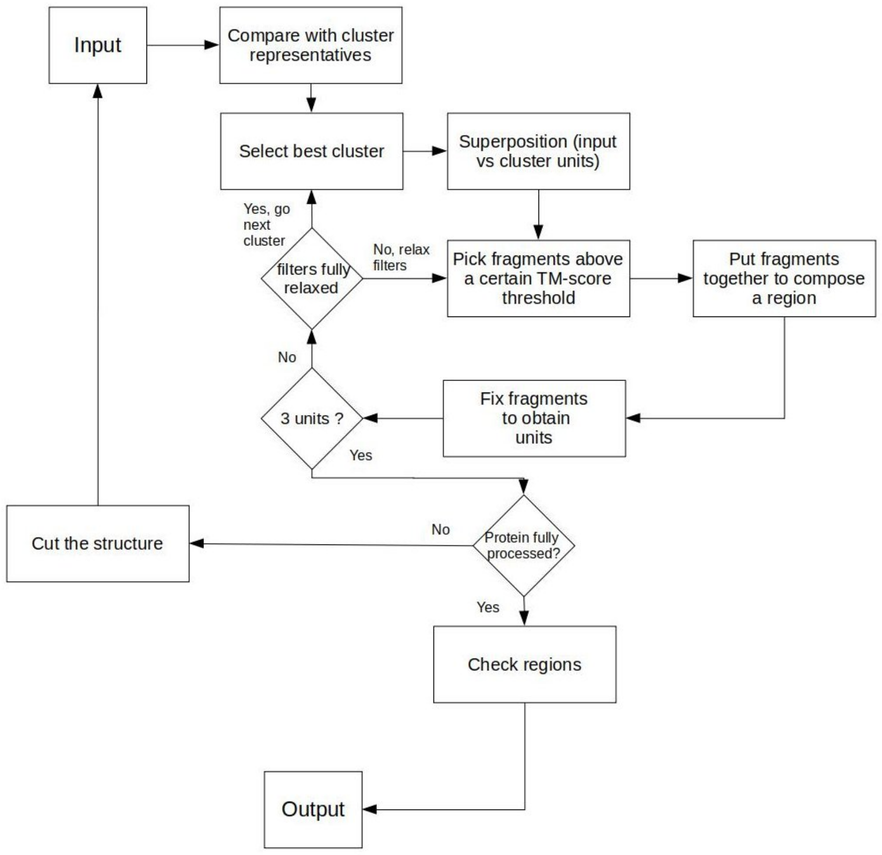
Input
Compare with cluster representatives
Select best cluster
Superposition (input vs cluster units)
Yes, go next cluster
filters fully relaxed
No, relax filters
Pick fragments above a certain TM-score threshold
Put fragments together to compose a region
No
3 units ?
Fix fragments to obtain units
Yes
Cut the structure
No
Protein fully processed?
Yes
Check regions
Output

Figure 4.1: schematic representation of the Algorithm

[†]https://biopython.org/

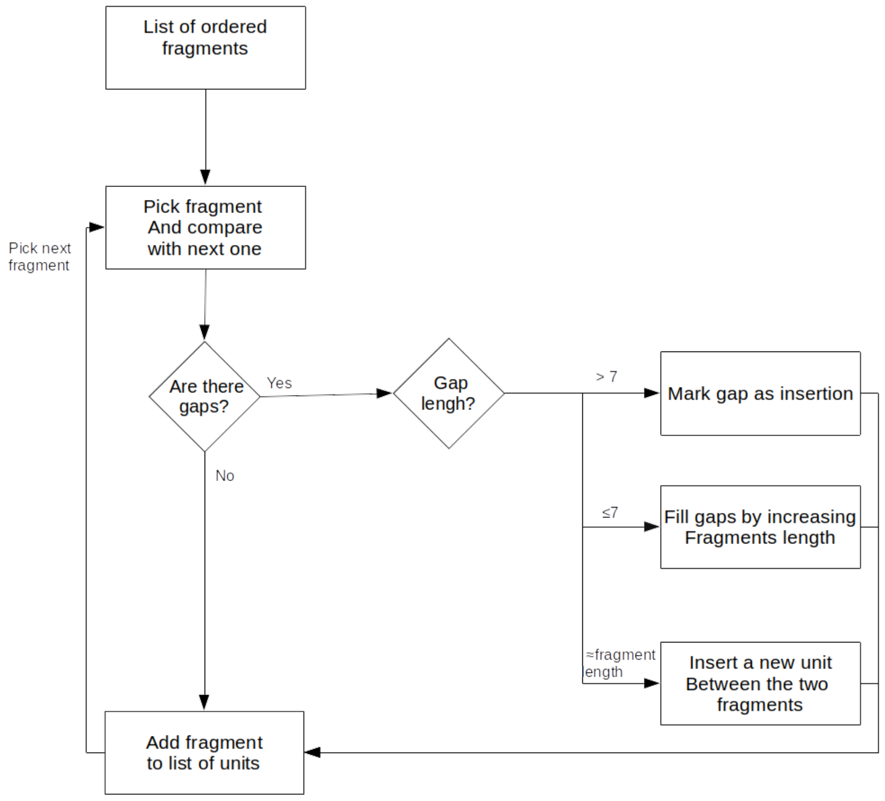
List of ordered fragments
Pick fragment And compare with next one
Pick next fragment
Are there gaps?
Yes
No
Gap lengh?
> 7
Mark gap as insertion
≤7
Fill gaps by increasing Fragments length
≈fragment length
Insert a new unit Between the two fragments
Add fragment to list of units

Figure 4.2: scheme representing how gaps between units are managed when fixing fragments

Tentative	iteration	TM-score	max gaps
1	1	0.30	5
	2	0.25	
	3	0.20	
2	1	0.30	10
	2	0.25	
	3	0.20	
3	1	0.30	15
	2	0.25	
	3	0.20	

Table 4.1: Set of parameters used by the algorithm. At each iteration if no solution is found the TM-score threshold is relaxed until a maximum of 0.20. If still no solution is found also the max gaps filter is relaxed.

PDB	region interval	predicted units	insertions	class	topology	fold	clan
6kivR	46-305	[[46, 86], [87, 129] ..]	None	Elongated	Propeller	..	..

Table 4.2: Example of output of the algorithm for a structure having one repeated region

4.2 Assessment methods

Validation is a very important step when designing a new algorithm because it can be useful to empirically establish how much your method is correct and if it performs consistently. Here, validation is intended as validation of the single units: indeed we want to check how good is the algorithm in predicting units boundaries. Unfortunately For tandem repeats proteins the situation is very complex because there is no model available for the majority of the repeats present in the world. For this reason the validation set will be only composed by some of the manually annotated proteins that are contained in RepeatsDB. The only way we can benchmark our prediction is by comparing them with the reference model available in RepeatsDB. As shown in figure 4.3 each unit of the prediction will be paired with one corresponding unit of the model that fits the best and if two units of the prediction can pair with the same unit of the model the one overlapping the most will be chosen [13]. After pairing units we have to compute the confusion matrix:

- **True positives** are the residues part of the same unit both in the model and in the prediction
- **True negatives** are the residues that are not part of any unit neither in the model or in the prediction
- **False positives** are the residues that are part of a predicted unit but not of the corresponding one in the model
- **False negatives** are residues contained in the model unit but not in the pairing predicted one.

The confusion matrix is the used to compute three fundamental metrics that are very common also in machine learning:

$$accuracy = \frac{TP + TN}{P + N} \quad precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN} \tag{4.1}$$

In our case they are useful because:

- Accuracy represents the fraction of residues our model was able to classify correctly
- Precision represents the amount of correctly classified residues among the ones that were assigned to units
- Recall identifies the amount of correct predicted residues in each unit.

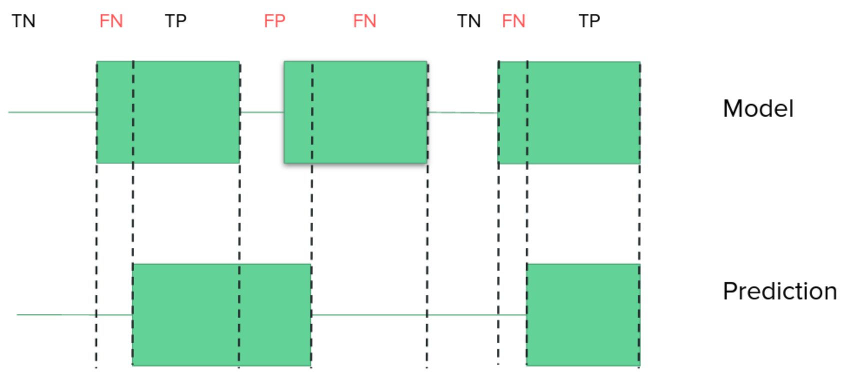

Figure 4.3: schematic representation of the validation procedure applied to two example regions. Units are represented as green blocks. Unit 1 of the model is paired with unit 1 of the prediction and unit 3 of the model is paired with unit 3 of the prediction. Hence, unit 2 is left out. True positives, false positives, true negatives and false negatives are highlighted

4.3 Prediction accuracy

Results were obtained by testing the algorithm on a dataset of 63 reviewed PDB selected randomly from RepeatsDB. Each PDB contains only one region for making evaluation easier and it is associated to one unique UniProt (no UniProt duplicates). Furthermore the units belonging to a region that was found in the input PDB were all discarded from the SRUL in order to force the algorithm to choose the master unit from other PDBs. We are interested in testing 3 different things using most of the metrics explained in 4.2.

- The accuracy, precision and recall in detecting units
- The precision and the recall in detecting repetitive regions.
- The correctness of the annotation given in terms of class, topology, fold and clan.

The full process of prediction of 63 PDBs lasted approximately 48 minutes. Only two PDBs out of 63 were labelled as ”not containing repetitive regions”.

class	number of regions
Beads-on-a-string	8
Closed repeats	27
Elongated repeats	28

Table 4.3: Number of regions associated to each RepeatsDB class

Table 4.3 shows how regions are divided per PDB. Elongated and closed repeats share a common number of candidates while beads-on-a-string are extremely lower.

	accuracy	precision	recall
region	77 %	92%	80%
unit	70 %	75 %	75 %

Table 4.4: Performance evaluation of RepeatsDB-lite 2.Measures are averaged over all repeat classes/topologies

Table 4.4 contains some general statistics about the results obtained: both precision and recall in detecting regions and units are encouraging. It is worth to notice that average unit detection match the one found in the validation set while recall is even higher. The average

class	region accuracy	region precision	region recall
Beads-on-a-string	75%	99%	75 %
Closed repeats	76%	92 %	80 %
Elongated repeats	77%	91 %	80 %

Table 4.5: Class precision and recall in detecting repetitive regions for RepeatsDB-Lite 2

topology	region accuracy	region precision	region recall
Alpha-barrel	97 %	98 %	99 %
Alpha-beads	92 %	99 %	93 %
Alpha-solenoid	87 %	99 %	88 %
Alpha/beta barrel	53 %	100 %	53 %
Alpha/beta prism	99 %	100 %	99 %
Alpha/beta sandwich beads	39 %	100 %	39 %
Alpha/beta solenoid	93 %	97 %	96 %
Alpha/beta trefoil	94%	97 %	97 %
Beta hairpin repeat	72 %	96 %	74 %
Beta-barrel / beta hairpins	50 %	100 %	50 %
Beta-beads	82 %	99 %	82 %
Beta-solenoid	58 %	78 %	63 %
Propeller	78 %	90 %	78 %
TIM-barrel	80 %	92 %	88 %

Table 4.6: Topology precision and recall in detecting repetitive regions for RepeatsDB-Lite 2

execution time per structure is a bit more than one minute, but we have to take into account that the RepeatsDB structures are usually smaller than the ones predicted by AlphaFold, so on real case scenario it will certainly require more time.

The precision and recall in detecting regions as represented in 4.5 are comforting. Closed and elongated repeats share same results, while beads-on-a-string have a lower recall. This probably due to the fact that most of beads-on-a-string have very long units that can be wrongly predicted as shorter and confused with other topologies belonging to closed and elongated repeats like beta-hairpin and alpha-solenoids.

Table 4.6, instead, show statistics about region precision and recall grouped by topologies. Most of the highly represented topologies like Propellers, TIM-barrels, alpha-beta solenoids and alpha-solenoids obtained good results. Other topologies like beta-barrels and beta solenoids even if quite present in the SRUL obtained worse results. It is possible that most of the can-

class	unit accuracy	unit precision	unit recall
Beads-on-a-string	68%	71 %	66 %
Closed repeats	69 %	80 %	76 %
Elongated repeats	71 %	70 %	76 %

Table 4.7: Class accuracy precision and recall in detecting units

topology	unit accuracy	unit precision	unit recall
Alpha-barrel	78 %	72 %	98 %
Alpha-beads	94 %	95 %	91 %
Alpha-solenoid	80 %	81 %	85 %
Alpha/beta barrel	16 %	51 %	35 %
Alpha/beta prism	85 %	88 %	95 %
Alpha/beta sandwich beads	28 %	41 %	18 %
Alpha/beta solenoid	85 %	76 %	93 %
Alpha/beta trefoil	88 %	90 %	97 %
Beta hairpin repeat	58 %	68 %	72 %
Beta-barrel / beta hairpins	43 %	87 %	46 %
Beta-beads	69 %	68 %	73 %
Beta-solenoid	58 %	56 %	58 %
Propeller	73 %	81 %	75 %
TIM-barrel	68 %	78 %	82 %

Table 4.8: Topology accuracy precision and recall in detecting units

didates contained in the SRUL are not quite good representative of those structural motifs. Moving on unit predictions by looking on table 4.7 we can notice that the class for which we obtained the best results are closed results with 80 % of precision. Elongated repeats are not that far but got 10 % less of precision. Beads-on-a-string got the worst results both in terms of accuracy and recall (as expected) since they are a minority in the SRUL.

In table 4.8 we group metrics results by topology to go more in depth in order to understand what were the structural motifs that got the most wrongly predicted: Alpha-solenoids, Propellers and TIM-barrels, alpha-beta solenoids are the most present both in the test dataset and in the SRUL and they got the best results. Beta-solenoid, instead, got not very good results probably because it is more difficult to detect unit boundaries and they can be easily confused with beta hairpins when using low TM-score threshold. Both beta-hairpins and beta solenoid are made of beta strands arranged in an antiparallel way, but in the case of beta hairpins they

classification level	correct annotations	wrong annotations
class	51	12
topology	48	15
fold	37	26
clan	36	27

Table 4.9: Amount of correct and wrong annotation per classification level

form an hairpin like structure while in the case of a beta solenoid they look like a superhelix. Alpha barrels, alpha beads and alpha/beta prisms all got good results but they should be tested on a wider sample: the number of regions in the dataset was not as representative as for example for TIM-barrels or Propellers. Beta-barrels, instead, are subject to the same problem of beta-solenoid but in a more serious way: beta-barrels, infact, are beta-sheet composed of beta-hairpins repeats.

Table 4.9 shows the number of correct and wrong annotations assigned to the regions. We expect class and topologies to be correct for most of the inputs and fold and clan to be more misspredicted. Results seems to confirm this tendency: for 12 PDB out of 63 the class level was missed and as a consequence also the more nested level of classification. However it is interesting to notice how the topology seems to be paired in the class in the sense that when the class is correctly predicted in most of the case also the topology is predicted (only in 3 cases the class is predicted and the topology is not predicted). The same reasoning is valid for fold and clan (for 36 regions out of 37 for which the fold was predicted the clan was guessed to). Plots 4.4 4.5 and 4.6 show the class, the topology and the fold of the regions that were annotated wrongly by the algorithm. By just looking at class we do not obtain many information: elongated repeats seems to be a bit more misspredicted in respect to beads-on-a-string and closed repeats but it could be related to the sample taken. The topology level highlights that many different topologies often belonging to different classes share common structural motifs: alpha-helices are at the same time units of alpha-solenoids but also of alpha-beads even if in alpha-beads alpha-helices are usually very long, beta beads on the other hand can be found smaller in beta-hairpins and longer in beta-beads. If, instead, we take a look at the fold level we see that the most wrongly predicted are alpha-solenoids and beta-solenoids. Indeed they can be often confused with alpha/beta-solenoids and in the case of beta solenoids even with beta-hairpin repeats. This reasoning makes clear how it is easy for an algorithm to assign wrong annotations: if we relax by a lot our TM-score and RMSD threshold it is way more probable that partially correct units can match with our input.

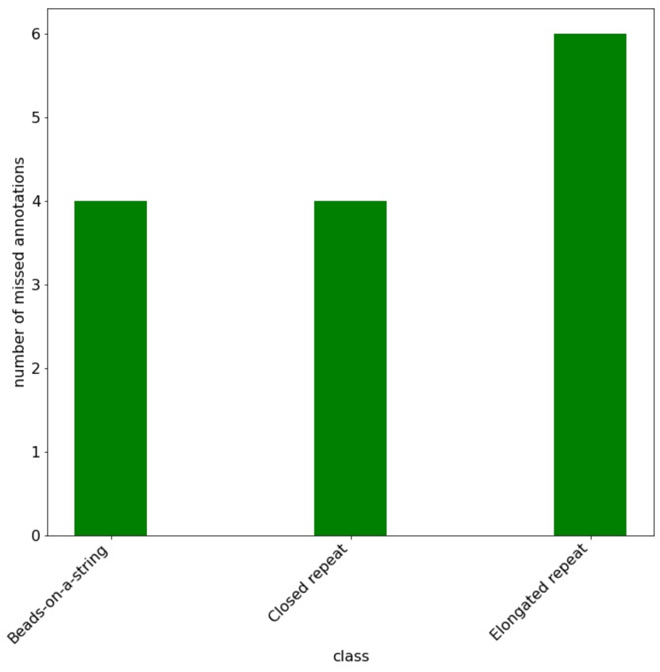

Figure 4.4: number of regions in each class annotated with a different class from the correct one

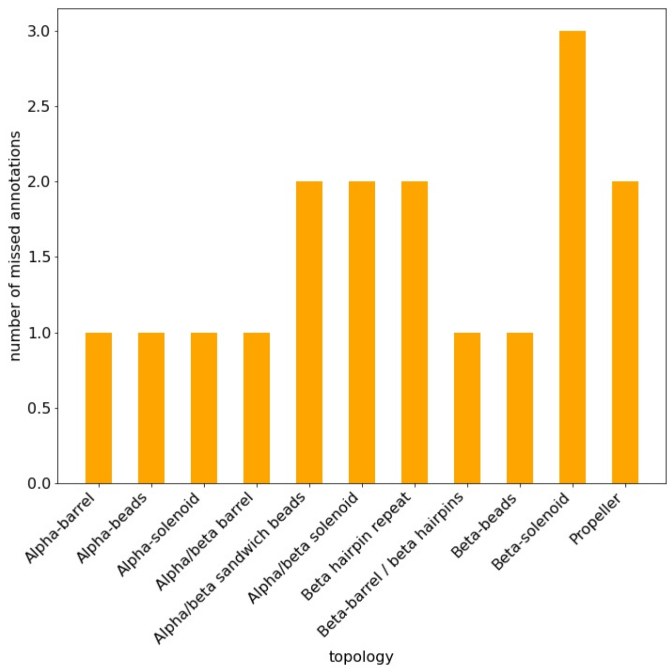

Figure 4.5: number of regions in each topology annotated with a different topology from the correct one

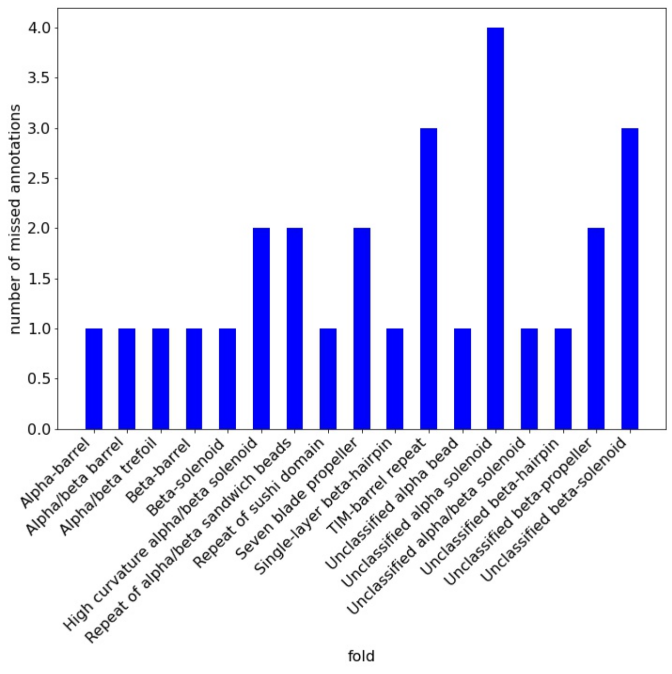

Figure 4.6: number of regions in each clan annotated with a different clan from the correct one

4.4 RepeatsDB-Lite and RepeatsDB-Lite 2: performances

In this section we are interested in making a comparison between RepeatsDB-Lite, the actual method used to annotate structures in RepeatsDB and RepeatsDB-Lite 2, the new algorithm. The 63 structures used to validate RepeatsDB-Lite2 were submitted to RepeatsDB-Lite which is freely available at `http://old.protein.bio.unipd.it/repeatsdb-lite/`. A first important thing to notice is the fact that RepeatDB-Lite was not able to identify repeats for 14 of the 63 structures unlike from RepeatsDB-Lite 2 were only 2 of 63 regions were not annotated. Moreover RepeatsDB-Lite, didn't discard from the SRUL units belonging to the input structure, hence this could affect predictions positively if those units are picked as masters. Table 4.10 show the average accuracy, precision and recall in detecting units and regions for RepeatsDB-Lite: precision and recall in detecting regions are overall extremely better for RepeatsDB-Lite but not in detecting units especially in accuracy and precision where the difference is at most 6 %: RepeatsDB-Lite 2 obtained a 70 % accuracy and a 75 % precision while RepeatsDB-Lite 76 % accuracy and 80 % precision. If we go more in details (table 4.11) and we compute metrics at a class level we can notice that beads-on-a-string and elongated repeats are overall better predicted by RepeatsDB-Lite, but for Closed repeat there are not important differences: precision is slightly higher for RepeatsDB-Lite 2 but accuracy and recall are similar or almost equal. Taking into consideration topologies as shown in table 4.12 we can make additional observations: alpha-barrel and alpha-beads, for example, are predicted better by RepeatsDB-Lite 2 (accuracy of 32 % against 78 % and 94 % against 70 %). For alpha-solenoids there are not many difference except for precision which is 7 % higher for RepeatsDB-Lite. Alpha/beta barrel and alpha/beta prism are overall better recognized by RepeatsDB-Lite even if also RepeatsDB-Lite achieved very good results. The same is valid for alpha/beta solenoids. Alpha-beta trefoil, on the other hand, are better predicted by RepeatsDB-Lite 2. Beta-beads, beta-hairpin repeat, beta-solenoid and beta-barrel/beta hairpins are predicted better by RepeatsDB-Lite: RepeatsDB-Lite seems more powerful in detecting beta-strands than RepeatsDB-Lite 2. On the other hand RepeatsDB-Lite 2, as highlighted before when analyzing metrics computed on classes is way better when dealing with closed repeats: accuracy and precision in detecting units is higher for both Propeller and TIM-barrel.

	accuracy	precision	recall
region	90 %	97%	92%
unit	76 %	80 %	98 %

Table 4.10: Performance evaluation of RepeatsDB-lite.Measures are averaged over all repeat classes/topologies

class	unit accuracy	unit precision	unit recall
Beads-on-a-string	81%	80 %	98 %
Closed repeats	70 %	73 %	83 %
Elongated repeats	81 %	87 %	85 %

Table 4.11: Class accuracy precision and recall in detecting units for RepeatsDB-Lite

topology	unit accuracy	unit precision	unit recall
Alpha-barrel	32 %	33 %	32 %
Alpha-beads	70 %	69 %	96 %
Alpha-solenoid	82 %	88 %	89 %
Alpha/beta barrel	98 %	98 %	100 %
Alpha/beta prism	97 %	97 %	100 %
Alpha/beta solenoid	91 %	87 %	96 %
Alpha/beta trefoil	83 %	84 %	94 %
Beta hairpin repeat	71 %	76 %	79 %
Beta-barrel / beta hairpins	65 %	83 %	75 %
Beta-beads	97 %	97 %	97 %
Beta-solenoid	72 %	93 %	66 %
Propeller	68 %	66 %	82 %
TIM-barrel	63 %	70 %	83 %

Table 4.12: Topology accuracy precision and recall in detecting units for RepeatsDB-Lite

4.5 RepeatsDB-Lite and RepeatsDB-Lite 2: execution times

RepeatsDB-Lite was a revolutionary and very well designed algorithm in the end. The main problem with it are its execution time: on average you require 8 to 10 minutes to process one RepeatsDB structure which is not feasible when you want to analyze thousand of PDB structures. For this reason i want to make clear that by implementing a new algorithm we had a radical improvement in terms of time performances. By submitting the 63 structures to RepeatsDB-Lite i was also able to estimate the execution times for each structure. Since RepeatsDB-Lite was executed from a webserver i removed 60 seconds from the final predictions to remove any overhead produced by the server answer. The total running time required RepeatsDB-Lite was 10 hours while the new algorithm was able to elaborate all inputs in just 40 minutes. Overall the difference is huge and also not expected. Figure 4.7 highlights in logarithmic scale how many

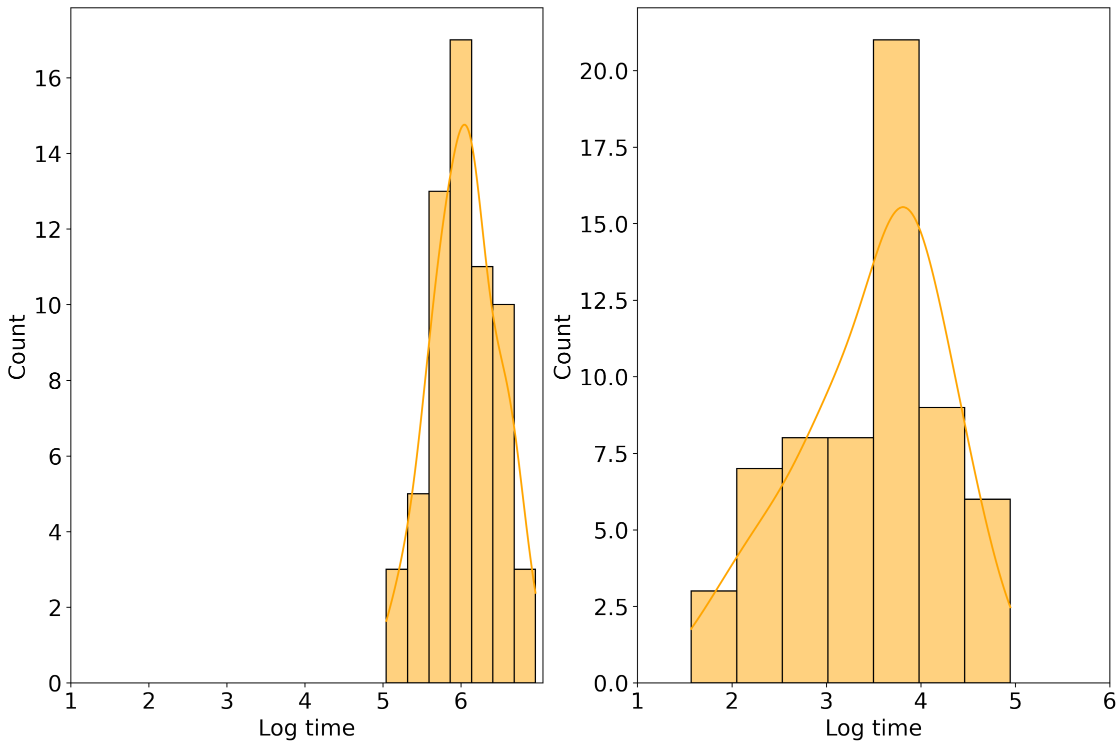

Figure 4.7: Histogram showing the big gaps in execution times between RepeatsDB-Lite (on the left) and the new algorithm (on the right) in logarithmic scale

seconds most of the predictions required. For RepeatsDB-Lite most of the structures required between 200 and 500 seconds (3 and 8 minutes) to be processed while for the new algorithm between 15 and 50 seconds. Overall, the difference is around one order of magnitude but we have to take into consideration a few additional things:

- RepeatsDB-Lite is not only a structural predictor but also a sequence based predictors. Its analysis are not limited to structural alignments
- The output produced by RepeatsDB-Lite is extremely more detailed and interactive than the one produced by our algorithm: RepeatsDB-Lite shows a multiple sequence alignments and also how units of the found regions align themselves
- RepeatsDB-Lite uses external software to make predictions and to plot results which can influence computational times in turn.

Implementing additional features on our algorithm would probably make execution times worse and more comparable to the ones of RepeatsDB-Lite, however preliminary results look very promising.

4.6 Large scale predictions

One of the main aims of structural bioinformatics is to develop new computational methods for the prediction of protein structures from the primary aminoacid sequence. Those predictors became fundamental mainly at the start of the XI century when the number of discovered sequences started to grow exponentially and became impossible to use the expensive and slow experimental techniques coming from molecular biology to determine for each sequence a corresponding structure. Since 1994 a biannual and community event called CASP (critical assessment of protein structure prediction) provided a way to different research groups to test the currently available software for secondary structure prediction. One of the most succesful results was obtained by Rosetta in 2004 when at 6th edition of the CASP conference was able to produce a close atomic level prediction with ab initio techniques (without using structural information coming from other structures). Ab initio predictors are considered the most difficult to design because they have to rely on just sequence information or on the modelling of physical interactions. Just for saying, RepeatsDB-Lite and the new algorithm explained in section **??** cannot be considered ab-initio methods since they both rely on a library of structural units. Rosetta starts by selecting from a library of sequence fragments the ones that match best with the input sequence. After that it combines the fragments and tries to produce a first model by optimizing non-local contacts. The process of optimization continues until the minimum of an energy function is reached by using the metropolis hastings algorithm. At the 13th edition of the CASP competition in 2018 a new algorithm developed by Google DeepMind and based on deep learning placed at the top rank. Its name is AlphaFold. AlphaFold [17] was considered a revolution in the world of structural predictors even if the full process leading to protein folding is still unknown, and it is still the standard secondary structure predictor nowdays. The first version of Alpha-fold uses an attention-based neural network which was modified for the second version that came out in 2020. AlphaFold 2 is based on a end-to-end neural network model for detecting residues structural relationships. It is an iterative procedure that after producing a raw model keep refining local interactions until a valid one is obtained. Molecular dynamics is used only in the final steps to refine the final predictions coming out from the neural networks. The importance of Alpha-Fold in the bioinformatics community just widened also to the world of tandem repeats where monthly new predicted proteins could expand our knowledge on repeats and contribute to enrich our currently available databases like RepeatsDB. Having new candidates structures is essential for template based methods: if we keep discovering new structure we can also generate a new improved (not necessarily bigger) version of the SRUL and make our algorithms better in detecting what before they were not able to spot. But it is not only this: keep testing algorithms on AlphaFold structures is a good procedure to understand if our current version of the SRUL contains undesidered too long or too short units that need to be filtered out.

4.6.1 Easy target

A first example i want to show is represented in figure 4.8. The protein represented is a proteins working as an antibody containing an immunoglobulin-like domain and found in hedgehogs. The structure shown was downloaded from the alphafold protein structure database and submitted to our algorithm [‡]. Even if no PDB data are associated to this structure, hence no units are represented in the SRUL the algorithm was able to detect two regions in 2 minutes and 44 seconds. The one on the left side of the picture (residues 84-474) was annotated as an alpha-beta solenoid and more specifically as a leucine rich repeats while the one on the right (residues 506-791) are considered beads-on-a-string (beta-sandwich beads). The validation of these structures is not possible because we have no PDB data associated to those structures, but we know specific structural topogies are often associated to specific functions so we can cross our information on structural repeats with the annotated domains in Pfam. Pfam is a database of protein families which seeks to provide a complete classification of protein families and domains. Figure 4.9 shows a comparison between the structural units found by the algorithm for the region 506-791 and the domains available in Pfam. It is evident that each of these structural repeats has a fundamental role in the function of the protein: the first domain ig_3 is a domain that is often involved in protein-protein interactions and protein-ligand interactions, while the second and the third domain are usually found in cell adhesion molecules where they play a fundamental role in the immune response.

[‡]https://alphafold.ebi.ac.uk/

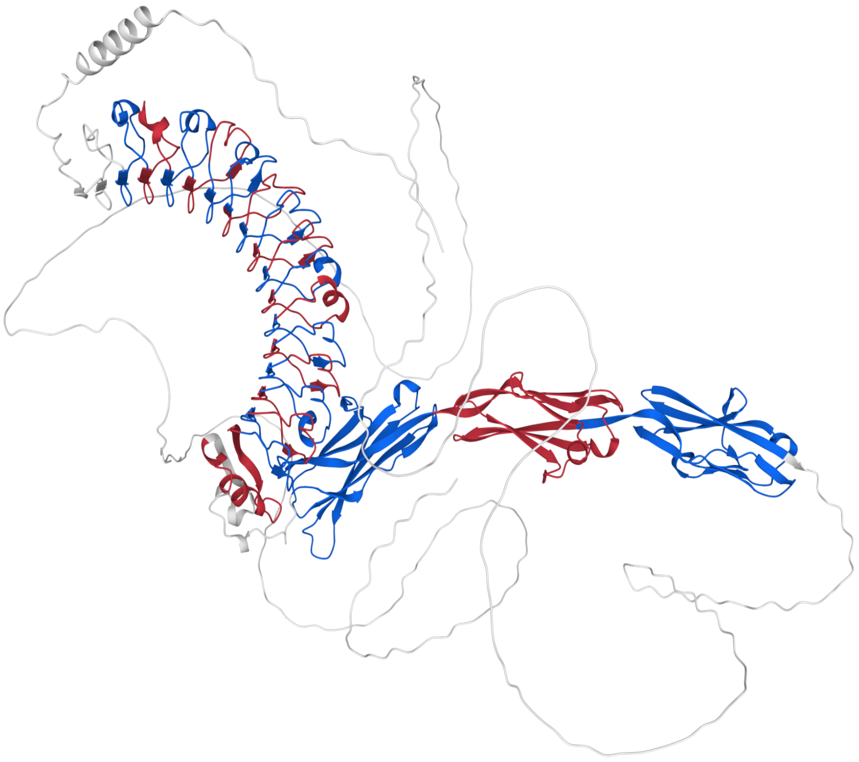

Figure 4.8: Alphafold predicted structure of UniProt A0A1S2ZKA9

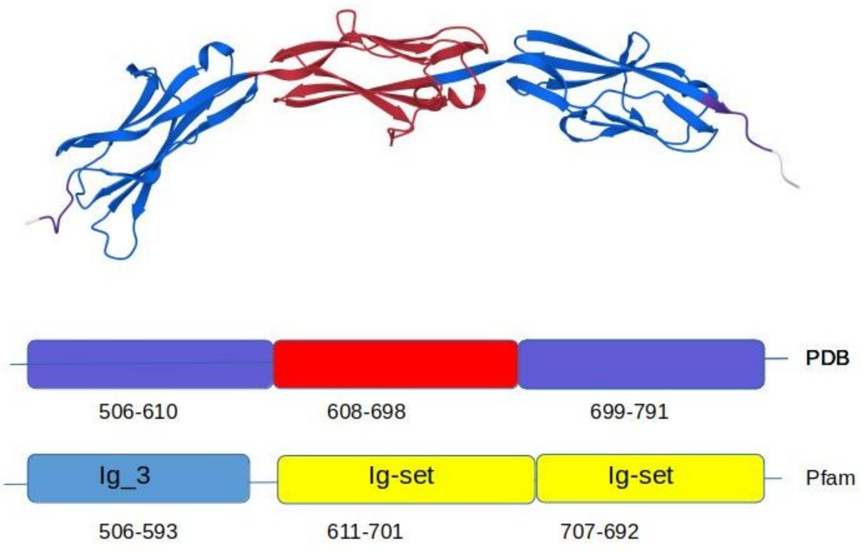

Figure 4.9: Comparison between structural domains found by the algorithm and information retrieved from Pfam

However often units do not form a domain on their own but they have to be assembled together. This is the case of region 84-474 which is shown in figure 4.10. The region is made of 16 units that are not forming a domain on their own: usually 2 or 3 units assembled together to form the so called leucin rich repeat (LRR) domain having a typical horsehoe shape with an interior array of beta-sheets and a external array of alpha helices.

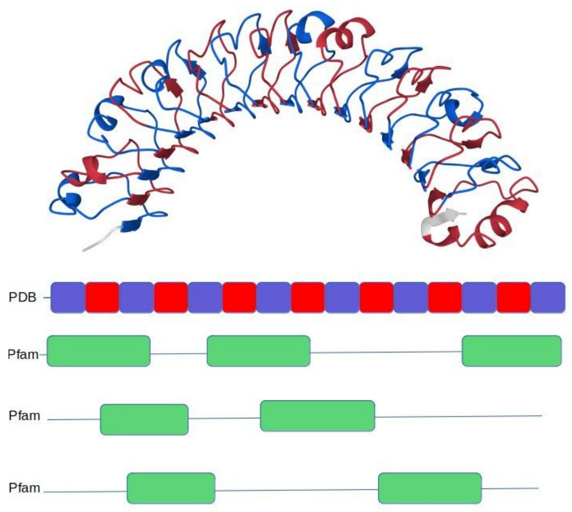

Figure 4.10: Comparison between structural domains found by the algorithm and information retrieved from Pfam

4.6.2 Difficult target

Another example i want to show is the AlphaFold structure related to UniProt Q8IXQ5 represented in figure 4.11. The algorithm was able to spot three different regions: the one of the right is a fibrous repeat, a coiled-coil for which no units are present in the SRUL. Indeed this region was annotated as an alpha-solenoid. Another region is a closed repeat and more specifically a seven blade propeller for which some units are present in the SRUL so it is easier to detect. The third region is something not expected and it is the one linking the propeller and the coiled coil. For this region which is an alpha-solenoid no PDB are available, so no repeated units associated with this region exist in the SRUL. The region was overall correctly spotted and annotated but the units are not: the first four helices are annotated as four independent units. This is a mistake because usually alpha-helices are made of two helices linked to each other having a length of at least 25 residues. This wrong prediction highlighted that the problem was one of the unit of the SRUL which was annotated as an alpha-solenoid but was just too short to be considered as such.

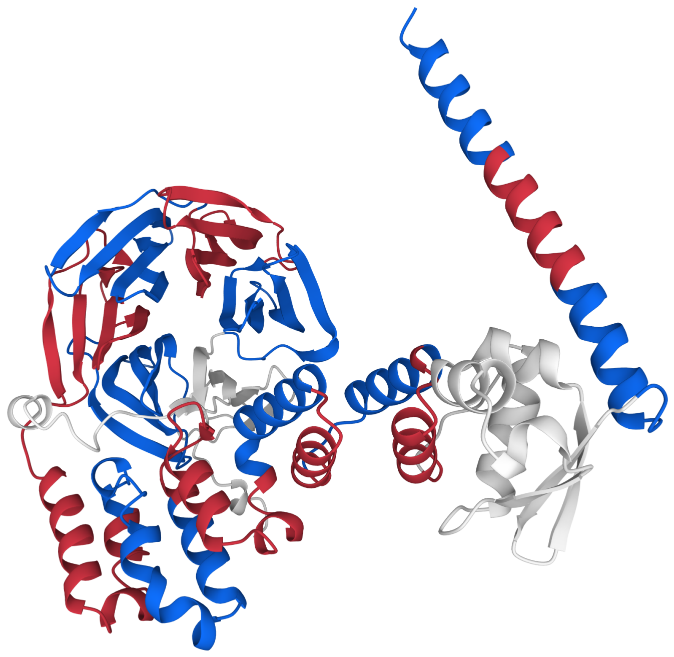

Figure 4.11: Prediction of the new algorithm for structure related to UniProt Q8IXQ5

5 Conclusion

We have developed an algorithm able to detect units inside repetitive regions and to annotate those regions using the classification of RepeatsDB. Two main aspects were curated: one is the Structural Repeat Unit Library containing a set of annotated units from RepeatsDB, the other is the algorithm itself. Many improvements can be made: The SRUL is certainly still not perfect, for example, oversampling or undersampling techniques could be explored to mitigate the elevate imbalance between classes and other clustering technique could be used. Developing a good algorithm for detecting repeats can put in front of many difficulties: most of the available predictors in the bioinformatics literature are not good at doing everything but they are biased and better in terms of performances towards specific classes or categories. It is important to be aware of the fact that when you are developing a computational method you are unconsciously putting some strong assumptions on the table and you are probably limiting your own algorithm when looking for the solution: in most of the cases it will work but in some other it will not, especially when you are dealing with biological entities that do not always obey to the strict and precise rules of mathematics. Even AlphaFold, one of the top tier automatic predictors in bioinformatics has its faults: its performances are great but when it has to deal with the full complexity of the world of proteins sometimes it fails. This is also something we have noticed for our new algorithm: some topologies were predicted better and some other not correctly detected. The issues here are doubled: you don't have to design only a fast, precise and correct algorithm but also to have a vast and comprehensive unit library that takes into consideration most of the variability present in nature. When dealing with biological data we widen our knowledge every day and we have to be continuously open to gather new data as soon as they become available. In order to test the correctness of our predictions we should rely on more than one predictor and build a good consensus because each predictor has its strengths and weaknesses. This is all to say that work here will never finish: the amount of structures available is still not representative of the thousand of proteins that are present in the environment and the computer will be even faster than today with higher amount of available memory and more speed. tomorrow there will be certainly more structure available, and as a consequence more chance to develop a faster and better-performing predictor.

References

[1] A. Kajava and S. C. Tosatto. (2018, Feb.) Editorial for special issue "proteins with tandem repeats: sequences, structures and functions". [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/29294402/

[2] A. V. Kajava. (2012) Tandem repeats in proteins: From sequence to structure. [Online]. Available: https://doi.org/10.1016/j.jsb.2011.08.009.

[3] T. D. Domenico, E. Potenza, I. Walsh, and et al. (2014) Repeatsdb : a database of tandem repeats unit structures. [Online]. Available: https://academic.oup.com/nar/article/42/D1/D352/1056634

[4] L. Holm and C. Sander. (1996) Mapping the protein universe. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8662544/

[5] I. N.Shindyalov and P. E.Bourne. (1998) Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. [Online]. Available: https://academic.oup.com/peds/article/11/9/739/1580846

[6] L. Holm and C. Sander. (1996) Protein structure alignment. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0022283689900843

[7] Y. Zhang and J. Skolnick. (April 2005) Tm-align: a protein structure alignment algorithm based on the tm-score. [Online]. Available: hhttps://doi.org/10.1093/nar/gki524

[8] L. H. Andreas Heger. (2000) Rapid automatic detection and alignment of repeats in protein sequences. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4585158/

[9] J. Jorda and A. V. Kajava. (2009) T-reks: identification of tandem repeats in sequences with a k-means based algorithm. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/19671691/

[10] L. Marsella, S. F. Sirocco, A. Trovato, F. Seno, and S. Tosatto. (2009) Repetita: detection and discrimination of the periodicity of protein solenoid repeats by discrete fourier transform. [Online]. Available: https://academic.oup.com/bioinformatics/article/25/12/i289/193063

[11] I. Walsh, F. G. Sirocco, G. Minervini, T. D. Domenico, C. Ferrari, and S. C. E. Tosatto. (2012) Raphael: recognition, periodicity and insertion assignment of solenoid protein structures. [Online]. Available: https://academic.oup.com/bioinformatics/article/28/24/3257/243769

[12] P. D. Viet, D. B. Roche, and A. V. Kajava. (2015) Tapo: A combined method for the identification of tandem repeats in protein structures. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0014579315007255

[13] L. Hirsh, D. Piovesan, L. Paladin, and S. Tosatto. (2016) Identification of repetitive units in protein structures with reupred. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/26898549/

[14] ——. (2018) Repeatsdblite: a web server for unit annotation of tandem repeat proteins. [Online]. Available: https://academic.oup.com/nar/article/46/W1/W402/4994211

[15] E. Krissinel. On the relationship between sequence and structure similarities in proteomics. [Online]. Available: https://academic.oup.com/bioinformatics/article/23/6/717/418732

[16] R. Bellman. (1957) Dynamic programming.

[17] J. Jumper, R. Evans, and A. P. et al. Highly accurate protein structure prediction with alphafold. [Online]. Available: https://www.nature.com/articles/s41586-021-03819-2

Acknowledgments

I would like to thank Professor Silvio Tosatto and the BioCompUp lab for the opportunity offered me and the interesting project they kindly proposed to me that perfectly suited my expectations. A special thanks goes to Professor Damiano Piovesan for his precious advices derived from interesting conversations and his unrelenting contribution in the development of this project. Finally i would like to thank also all people that supported me in the last years.