UNIVERSITY OF PADUA

MASTER THESIS

Contacts prediction of linear peptides from genomic data

Author:
Damiano CLEMENTEL

Supervisor:
Prof. Damiano PIOVESAN

Co-Supervisor:
Prof. Silvio TOSATTO

A thesis submitted in fulfillment of the requirements for the degree of Data Science

in the

Department of Mathematics
Data Science

April 29, 2021

Abstract of thesis entitled

Contacts prediction of linear peptides from genomic data

Submitted by

Damiano CLEMENTEL

for the degree of Data Science

at The University of Padua

in April, 2021

The rise of metagenomics and the technological improvements in the fields of bioinformatics and computational biology led to an exponential increase in the amount of biological data available to be studied. However, the rate at which biological data are studied is much slower than the rate at which they are stored.

This issue pushed the development of programs capable of extracting significant information from newly sourced data without the need of human intervention. More specifically, some of these programs have been developed to infer structural information from protein sequences. Since the structure of a protein is strictly bound to its function, it is easy to understand the importance of such task.

Among the structural information which can be inferred looking at a protein sequence, there are contact maps. Contact maps define whether two residues are functionally linked within the same protein chain or two different ones.

Despite much work has been carried out for intra-chain contact maps prediction using sequence information, less can be found about inter-chain contact maps. Moreover, methods are usually presented and tested on benchmark dataset generated for such purpose.

In this, a whole pipeline for both intra-chain and inter-chain contact predictions is presented. Instead of using a generic benchmark set of protein sequences as input, the pipeline starts from predictions of linear interacting peptides at residues level.

Linear interacting peptides are regions in a protein sequence which are thought to not have a fixed folding, but to adapt their structure to the functional needs of the protein itself. Needles to say, fewer studies have been conducted about this specific issue in literature.

Finally, an analysis of the results is carried out. The analysis focuses on the evaluation of methods implied for contact predictions over the given dataset. Particular attention is paid to the comparison of the performances on inter-chain alignments with respect to the ones achieved on intra-chain alignments. Furthermore, the effect of linear interacting peptides is taken into account.

Contacts prediction of linear peptides from genomic data

by

Damiano CLEMENTEL
B.S. *University of Trento*
M.S. *University of Padua*

A Thesis Submitted in Fulfilment
of the Requirements for the Master's Degree in
Data Science

at

University of Padua
April, 2021

Acknowledgements

I would like to thank Prof. Tosatto and the whole BioCompUP laboratory for the opportunity offered to me. Special thanks go to Prof. Piovesan for his time and patience in the supervision of this work and for all his useful discussions. Thanks also to Dr. Micetic, for the helpful technical advice.

Moreover, I would like to thank my whole family, especially my parents and my sister who have always supported and encouraged me during my entire academic career so far. Thanks also to Sara for always being by my side when needed an to her family as well.

Last but not least, I am particularly grateful to my friends Domenico, Francesco, Laura, Martino, Nicolò and Paolo for the effort put in answering all of my questions and for never miss the chance to offer their valuable point of view.

Damiano CLEMENTEL
University of Padua
April 29, 2021

Contents

Abstract i

Acknowledgements ii

List of Figures v

List of Tables vii

List of Abbreviations ix

1 Introduction 1
- 1.1 Introduction to proteins . . . 2
- 1.2 Properties of amino acids . . . 5
- 1.3 Sequence similarity and homology searching . . . 7
- 1.4 Linear Interacting Peptides (LIPs) . . . 10
- 1.5 Tools for searching sequence databases . . . 12
  - 1.5.1 BLAST . . . 12
  - 1.5.2 PSI-BLAST . . . 13
- 1.6 Databases for biological entities . . . 14
  - 1.6.1 UniProt Knowledgebase . . . 15
  - 1.6.2 Protein Data Bank . . . 17
  - 1.6.3 Structure Integration with Function, Taxonomy and Sequences . . . 19
- 1.7 Tools for protein contacts prediction . . . 21
  - 1.7.1 Covariance . . . 23
  - 1.7.2 Shannon's entropy and mutual information . . . 25
  - 1.7.3 Decomposition of mutual information sources . . . 28
  - 1.7.4 Mutual information normalization . . . 29
  - 1.7.5 Mutual information correction for phylogeny . . . 30
  - 1.7.6 PSICOV . . . 34

2 Methods 37
- 2.1 Application of mutual information to multiple sequence alignments . . . 37
  - 2.1.1 Covariance . . . 38
  - 2.1.2 Shannon's entropy and mutual information . . . 39
  - 2.1.3 Pseudocount for low number of sequences . . . 41

2.1.4 Entropy normalized mutual information . . . 41
2.1.5 Phylogeny corrected mutual information . . . 41
2.2 Pipeline . . . 42
2.2.1 Initialization of output dataset . . . 43
2.2.2 Linking residue-level information between external resources . . . 45
2.2.3 Distance matrices calculation . . . 47
2.2.4 Multiple sequence alignments generation . . . 50
2.2.5 Hobohm clustering algorithm . . . 51
2.2.6 Prediction of intra-chain and inter-chain contact maps . . . 53
2.2.7 Generation of summary tables . . . 54

3 Results **57**
3.1 Retrieved linearly interacting protein structures . . . 57
3.2 Choice of the best threshold for prediction methods . . . 68
3.2.1 Choice of threshold values for mutual Information . . . 69
3.2.2 Choice of threshold values for PSICOV . . . 75
3.3 Evaluation of contact prediction methods . . . 79

4 Conclusions **83**

Bibliography **85**

List of Figures

1.1 Quaternary structure of human Hemoglobin, with PDB identifier 1A3N.[22] 3
1.2 Amino acid scheme [https://it.wikipedia.org/wiki/Amminoacido] . . . 5
1.3 Amino acid chain with dihedral angles ϕ and ψ 6
1.4 An hypothetical phylogenetic tree. All genes descend from a common ancestor 8
1.5 Relationship between protein sequence, structure and function. [15] . . . 10

3.1 Number of items in initial dataset 58
3.2 Distribution of the resolution of the protein (PDB) structures. 58
3.3 Distribution of the number of chains in each PDB structure 59
3.4 Distribution of the number of chains for each UniProt entry 60
3.5 Distribution of the width for protein chains, also called chain width. . . 61
3.7 Distribution of the size of input alignments. 63
3.8 Number of output obtained through different contact prediction methods 65
3.9 Distribution of the width of the alignments 66
3.10 Alignments height distribution. 67
3.11 Distribution of the Z-scores for different prediction methods applied on the training dataset. 70
3.12 Correlation matrix between both the features of the input alignments and the statistics of mutual information. 71
3.13 Variation of average standardized mutual information with the width of the alignment. 72
3.14 Distribution of average mutual information for intra-chain and inter-chain alignments, separately. 72
3.15 Mean precision, sensitivity and F1-score for contact prediction methods based on mutual information applied on the training set. 73
3.16 Correlation matrix between the features of the input alignments 76
3.17 Distribution of the target contact map density computed between carbon atoms or heavy atoms, conditioned by the the type of input alignment . 76
3.18 Distribution of the positive predicted values. 77
3.19 Distribution of the positive predicted values. 77
3.20 Distribution of precision, sensitivity and F_β-score on the test set 79
3.21 Distribution of precision, sensitivity and F_β-score on inter-chain alignments of the test set 80

List of Tables

1.1 Summary alignments table 22

2.1 Matrix defining the identifiers of both inter-chain and intra-chain alignments between reference chains in the PDB structure 1cxp. 47
2.2 First 10 residues mapped between PDB, UniProt and LIP predictions for chain A in PDB structure 3m91 48
2.3 Atom coordinates for the first 10 residues mapped between PDB and UniProt for chain A in PDB structure 3m91 48
2.4 Atomic distances between the 7th and 10th residue in chain A of the PDB structure 3m91 50
2.5 Example of Hobohm-1 algorithm result. 52
2.6 Example of percent identity computation. 53
2.7 Summary chains table 54
2.8 Summary alignments table 55

3.1 Best parameters for methods based on mutual information, computed on training partition 74
3.2 Best parameters for PSICOV. 78

List of Abbreviations

MSA	**M**ultiple **S**equence **A**lignment
LIP	**L**inear **I**nteracting **P**eptides
IDP	**I**ntrinsically **D**isordered **P**rotein
IDR	**I**ntrinsically **D**isordered **R**egion
PDB	**P**rotein **D**ata **B**ank
BLAST	**B**asic **L**ocal **A**lignment **S**earch **T**ool
PSI-BLAST	**P**osition-**S**pecific **I**terative **BLAST**
SIFTS	**S**tructure **I**ntegration with **F**unction, **T**axonomy and **S**equence
PSICOV	**P**recise structural contact prediction using **S**parse **I**nverse **COV**ariance estimation

Chapter 1

Introduction

In this chapter, some biological notions strictly related to the task of contact predictions are explained. This includes the definition of what a protein is and the reason for which it is a key element in any living organisms. Other concepts strictly related to proteins such as amino acids and sequence homology are taken into account, as well.

The generation of multiple sequence alignments is a key step in contact prediction and is closely related to sequence homology. Hence tools involved in the retrieval of such resources are introduced below and a particular attention is paid on their algorithmic aspect. The first tool introduced is BLAST. It is a pairwise sequence search an alignment tool. Despite not being used in the pipeline described in Chapter 2.2, it is important to understand PSI-BLAST. The latter is an enhancement of BLAST and is the tool actually used in the pipeline to produce multiple sequence alignments.

Another important element in the prediction of contacts at residue level are the datasets. In this section, thee databases are described. The first one is the Protein Data Bank (PDB). It contains three dimensional structure of the proteins to be analyzed. The second one in UniProt, which contains biological sequences, instead. The third one is Sequence Integration with Function, Taxonomy and Structure. It plays a key role in mapping resources between the former two databases.

1.1 Introduction to proteins

Proteins are biological macromolecules made of a large number of smaller molecules called amino acids, which are the building blocks of the protein. Amino acids composing a protein are organized in one or more long sequences, known as protein chains or simply chains. The amino acidic sequence of each protein chain forms the primary structure of the protein itself. The primary structure of a protein can be represented as a sequence of characters. Each position in such sequence holds an amino acid, which is encoded with a specific character. Every amino acid is uniquely identified by one character, also called one-letter-code. There are 20 different one-letter-codes, 20 being the number of the amino acids in the standard genetic code.

The function of the proteins is of key importance for the metabolism of the cells in living organisms. Most of them carry out enzymatic functionalities, hence the increase in the rate of specific chemical reactions through catalysis. However, proteins are involved in other tasks as well. Some proteins have a structural function, since they are involved in maintaining the shape of the cell. Some others have a mechanical function, since they influence the contraction ability of the muscles. Moreover, proteins take part in the duplication process of the cells, play a key role in the immune response and are involved in both the communication and the interaction between different cells as well.

The functionality of a protein is associated to its overall physical and chemical properties, which are bound to the three dimensional conformation it assumes in the three dimensional space, also known as its folded structure. Given a protein chain, its structure is mainly influenced by the physical and chemical properties of the amino acids which compose it, hence its primary structure, and by the physical interactions of such amino acids between each other and with the other chains inside the same protein.

It is possible to associate a sub-sequence within the primary structure of a protein chain to a specific, well known three dimensional structure, called structural motif. The set of all the structural motifs internal to all the chains in a protein makes up its secondary structure. Among said three dimensional sub-structures, the most common ones are α-helices and β-strands, each with different characteristics which influence the functionality of the protein itself. As suggested by the name, the amino acid strand in an α-helix is arranged in a spiral shape. This implies that the sites in its primary structure are in contact with the ones coming three to four positions earlier. In a β-strand, instead, only a few amino acids are arranged in a stretch. More than one of those strands connected together by a turn motif such as a β-turn and positioned on the same three dimensional plane compose a β-sheet. Moreover, β-sheets are thought to be involved in fibrils, which in turn are related to human diseases such as Alzheimer.

By taking a higher level look at the folded structure of a single protein chain and taking into account its interactions with other chains in the same protein, we get its tertiary structure. Quaternary structure considers the whole protein structure and its

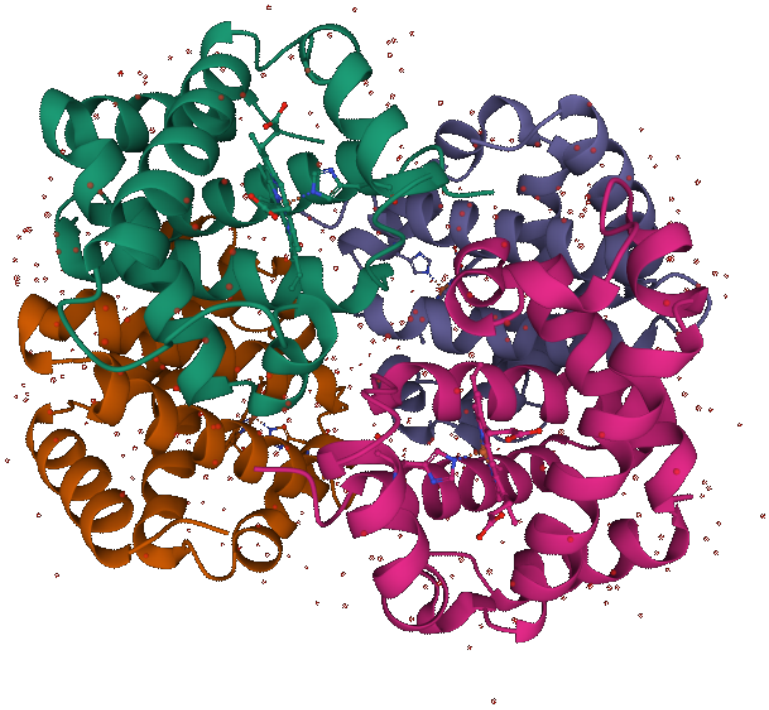


Figure 1.1: Quaternary structure of human Hemoglobin, with PDB identifier `1A3N`.[22]

interactions with other proteins, instead. An example of quaternary structure can be found in Figure Fig. 1.1. Protein chains are clearly distinguishable by their color.

Proteins loss and production inside every cell is balanced through a process called protein biosynthesis. Such process starts from DNA strands, obtained by splicing one DNA helix, which can be found in each living organism, being it either prokaryotic and eukaryotic. Simplifying, a DNA strand can be considered as a sequence of molecules called nucleotides, each of which is composed and identified by a nucleic base: cytosine (C), guanine (G), adenine (A) and thymine (T). Sub-sequences in DNA strands which encode for a protein are known as genes. The first step of protein biosynthesis is gene transcription indeed, and takes places inside the nucleus of the cell. During this phase, a gene is transcribed into a mRNA strand by means of mRNA polymerase enzyme. Last step is the synthesis of mRNA sequence into the protein sequence itself, which is carried out by ribosomes.

Once protein primary structure has been synthesized, it must fold to the correct three dimensional structure in order to carry out its functionality as expected. Such conformation is known as the netive fold of the protein. Protein folding happens spontaneously, mostly driven by physical forces arising from the properties of the amino acids which compose its primary structure, including hydrophobic interactions, Van

der Waals forces and hydrogen bonds. For example, a protein being surrounded by water molecules will cause hydrophobic amino acids to move to the internal core of the protein structure itself, while being shielded by hydrophilic ones. Moreover, some proteins called chaperones are capable of helping other proteins to fold and to correct folding errors. However, it is possible that issues happen either during protein biosynthesis and folding anyway, resulting in wrongly folded proteins. This, in turn, can possibly lead to the death of the cell and consequently to diseases of the organism to which they belong.

Theoretically, it should be possible to find the exact fold of a protein by considering the thermodynamic energy function linked to its primary structure and to the physical properties of its components. Therefore, the native conformation of the structure of the protein can be computed by finding the minimum value of such function. Various methods that solve this problem through an optimisation algorithm are available and classified as *de-novo* protein structure prediction methods. In contrast, *ab-initio* models predict the structure of the protein by first finding sequences with similar primary structure and whose three dimensional structure is known, then optimize its fold a posteriori.

1.2 Properties of amino acids

An amino acid is a molecule whose atoms are structured in three different functional groups, namely an amino terminus, a carboxyl terminus and a side chain. While the first two are ever the same for each amino acid, the side chain can change and its composition gives the name to the amino acid itself.

The amino terminus, also known as N-terminal end, is composed of two atoms of hydrogen (H) bound to the same nitrogen atom (N) forming an amine group ($-NH_2$). The carboxyl terminus, or C-terminal end, is a carboxyl group ($-COOH$) instead. Therefore, it is composed of one carbon atom (C) bound to two different oxygen atoms (O), among which one is bound to an hydrogen atom. Carbon atom inside the carboxyl terminus is commonly referred to as beta (β) carbon.

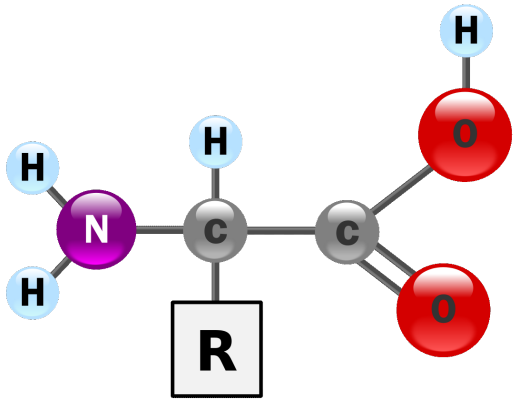


Figure 1.2: Amino acid scheme [https://it.wikipedia.org/wiki/Amminoacido]

In proteinogenic amino acids, the nitrogen atom inside the amino terminal is attached to a carbon atom, which is in turn attached to the carbon atom inside the carboxyl terminus. The carbon atom connecting both amino and carboxyl terminus is called alpha (α) carbon and it plays a key role in the structure of the amino acid, since it bonds with another hydrogen atom and with the side chain as well. All atoms involved in the amino acid structure are attached together by means of covalent bonds.

By convention, functional groups in amino acids are read left to right, with the N-terminal end on the left hand side of the alpha carbon atom attached to the side chain. On its right hand side the C-terminal end is found instead. Two adjacent amino acids can be bound together through a peptidic bond between the carboxyl terminus of the first one and the amino terminus of the second one. Peptidic bond happens when the diatomic molecule composed of oxygen and hydrogen in C-terminal end reacts with an hydrogen atom in the N-terminal end, producing one water molecule H_2O. As a consequence of this reaction, the carbon beta atom in C-terminal end of the first amino acid and the nitrogen atom in N-terminal end of the second amino acid result to be attached together by means of a covalent bond. When a few amino acids are attached

together in sequence through peptidic bonds, the whole molecule can be classified as peptide. Instead, if the amino acid involved are many, for example in a protein chain, it can be classified as a polipeptide.

The physical and chemical properties of an amino acid, along with his name, are associated to its side chain. As we already mentioned, in proteinogenic amino acids their side chain is attached to alpha carbon atom, therefore they are classified as alpha amino acids. According to its side chain composition and to the solution in which it is measured, the amino acids can have either have a positive charge, a negative charge, be polar or nonpolar. Since a water molecule is polar, an amino acid is considered hydrophilic when it is polar as well, since its positively charged pole will attract the negatively charged pole of the water molecule and vice versa. A nonpolar amino acid is considered as hydrophobic instead. Moreover, by measuring the electric charge of an amino acid in a solution at physiological pH, is is possible to classify them in five different groups.

The three dimensional structure of protein chains is influenced by angles occurring inside each amino acid between its N-terminal end and its side chain and between its side chain and its C-terminal end, as well as between its C-terminal end and the N-terminal end of the next amino acid. Such angles are called, phi (ϕ), psi (ψ) and omega (ω) respectively. Since the peptide bond connecting two amino acids together is planar, the latter is a planar angle, hence its value is constrained to be either 0 or 180 degrees. Instead, Phi and psi angles are strongly influenced by the composition of the side chain. Only a few of their combinations have a significant probability of being found in nature. Moreover, it has been shown that the value assumed by those two angles in a polipeptide strongly correlates with its motif, i.e. its three dimensional local conformation. [1]

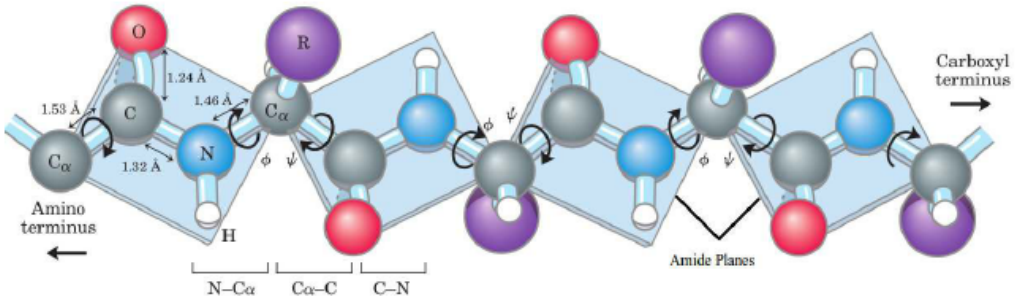


Figure 1.3: Amino acid chain with dihedral angles ϕ and ψ

1.3 Sequence similarity and homology searching

Living organisms are constantly pushed to adapt to their environment by evolutionary forces. Among these forces there are mutation and genetic recombination. Both mutation and genetic recombination involve changes in the genome which lead to changes in the phenotype as well. Hence, those individuals showing a characteristics particularly suitable to the environment they live in will be favoured over those not showing it.

Looking at the adaptation phenomenon under the point of view of genetic sequences, changes in the genome result in changes in their product, being it a sequence of mRNA or a protein. Those changes can then accumulate as evolution and time goes on. When a sufficient number of changes in the genetic material accumulated as to produce an observable difference in some individuals, then a new species arise.

Different biological sequences, being them DNA, RNA or proteins, are said to be homologous when they share a common ancestor sequence. Homology can have shared ancestry because of different reasons. Among those reasons, the most relevant ones are orthology and paralogy. The concept of homology in protein sequences in strictly related to the concept of co-evolution.

Orthology occurs when two sequences share the same common ancestor after a speciation event. As the name suggests, in a speciation event two populations diverges the one from the other, each one generating a new species. Despite being two different species, these species still have many genes in common. These genes and their products are then said to be orthologous.

Instead, paralogy is intended as homology due to a duplication event. When a gene is duplicated in the last common ancestor, both the original and the duplicated ones will mutate separately. After a speciation event, both the paralogous genes will be carried on to the different species. Therefore, two protein sequences are said to be paralogs when they are the product of two genes which originated from the same one in the last common ancestor.

In the biological field, orthology is supposed to provide more functional similarity than paralogy, while is still discussed. This means that the same gene carried on from last common ancestor to two different species should fulfil the same function, despite being it mutated independently form the other due to genetic forces. Since the function of protein synthesised from the gene is strictly bound to its native fold, structural similarity can be inferred through orthology as well. [14]

Sequence homology is strictly bound to sequence similarity. However, these two terms do not indicate the same concepts, nor one can be used in substitution of the other. While there are homologous protein sequences which are not similar, it is possible to use similarity to infer homology. Instead, it is not possible to infer whether two sequences are not homologous even if they are different.

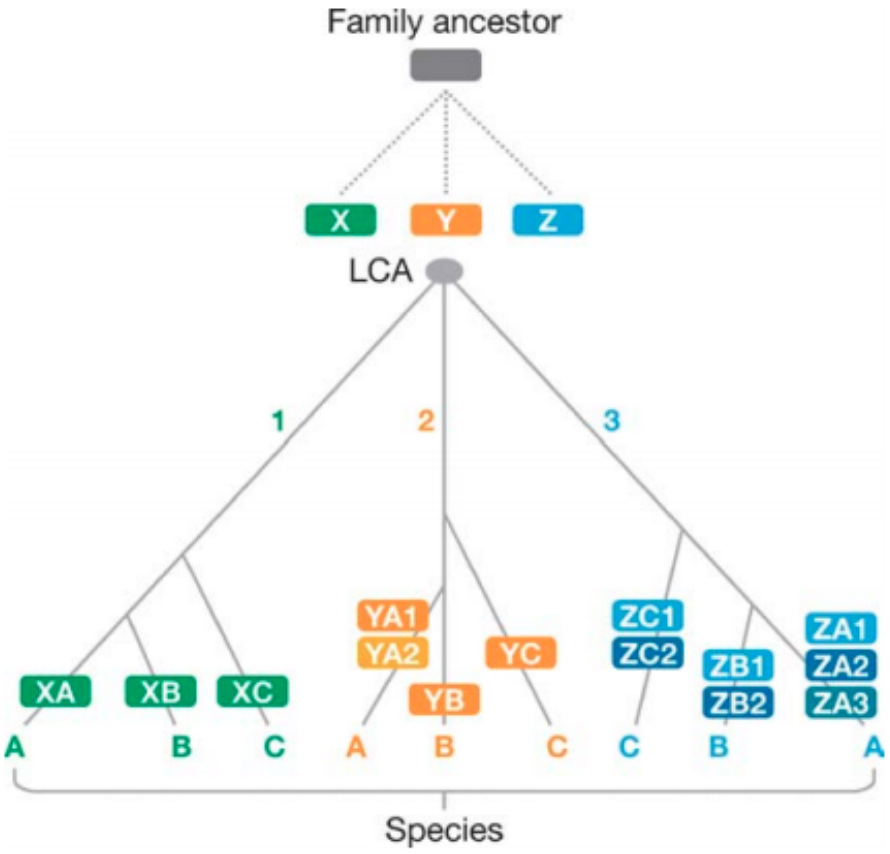


Figure 1.4: An hypothetical phylogenetic tree. All genes descend from a common ancestor

Similarity search tools like FASTA, PSI-BLAST and HMMER have been developed to search for homologous sequences. The first two, namely BLAST and PSI-BAST, have been described in Section 1.5. The principle upon which these tool are based to infer homology is the excess of similarity. Therefore, homology is assessed if the observed similarity is significantly higher than the score which would be expected by chance.

Usually, pairwise similarity search tools produce a similarity score whose distribution is explained by Equation (1.1), like in the case of FASTA and BLAST. However, these tools usually rely on a bit-score, rather the raw score. Bit-score does not suffer from normalization issues which affect the raw score, instead. Moreover, bit-score can be easily converted to a probability by means of Equation (1.1), with m, n the lengths of the sequences involved.

$$p(s \geq x) = 1 - \exp(-\exp(-x)) \tag{1.1}$$

$$p(b \geq x) = 1 - \exp(-mn2^{-s}) \tag{1.2}$$

The tools reported before are not intended to search whether a query sequence

is homologous to single target sequence, rather to a lot of target sequences stored in database such as UniProt, described in Section 1.6. Hence, it is important to find out which are the sequences more likely to be homologous. These information can be found through the expectation value, usually referenced as E-value. Being b the bit score obtained comparing two sequences, the expectation value is defined as the number of times such bit-score is expected to be observed by chance in the whole target dataset. Given D the number of all target sequences, the $E - value$ can be computed as follows:

$$E(b) \leq p(b)D \tag{1.3}$$

The introduction of the multiplier D in the computation of the score reported by the sequence similarity search tools, makes it dependent form it. This means that the same score retrieved by searching against a large dataset will be less significant than the same score retrieved from a smaller dataset. Then, if a target sequence is found to be homologous to a query sequence in the smaller dataset, it is not granted that such homology will persist in the larger one. Therefore, it should be easier to infer homology correctly in the former, with respect to the latter. [19]

1.4 Linear Interacting Peptides (LIPs)

As already mentioned in Section 1.1, the function of a protein is strongly related to its folding, hence its three dimensional structure. It has been already discussed how both the tertiary and the secondary structures of a protein can be inferred from its primary sequence. Furthermore, this relationship between a protein sequence and its structure can be exploited to determine the structure of others with a similar sequence. [15]

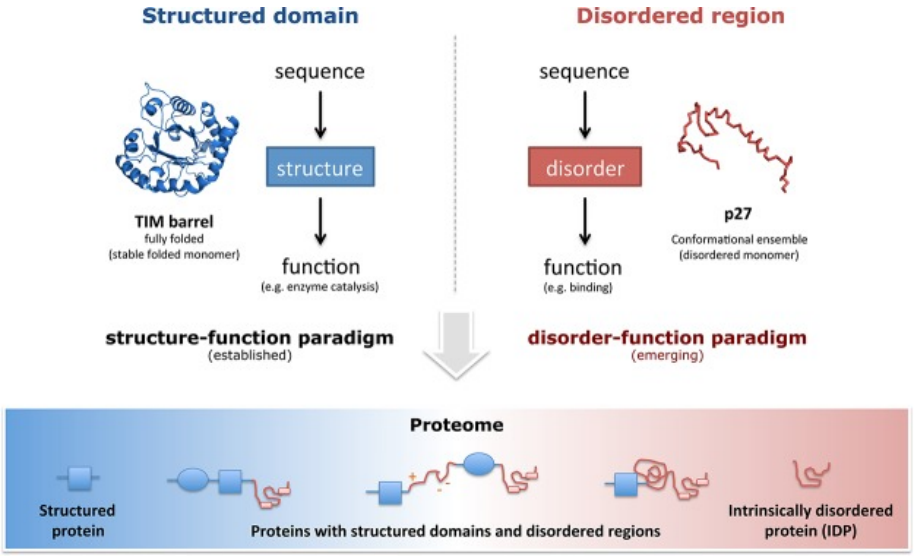


Figure 1.5: Relationship between protein sequence, structure and function. [15]

However, some proteins or protein regions are not able to fold in a globular structure due to the physical and chemical properties of their amino acids. Protein regions or proteins not associated to any specific folding are known as intrinsically disordered regions (IDR) or proteins (IDP), respectively. Intrinsically disordered regions were thought to be useful only as a connection between two structured regions.

Recently, new functionality has been associated to intrinsically disordered regions. Indeed, these regions provide to the protein with the ability to change its structure and adapt to a binding site. Hence, their fold is no more fixed but changes over time. Disordered regions usually miss an hydrophobic core and assume a linear structure when they are not bounded to anything. Instead, their fold changes when bounded to some specific molecule. Therefore, they are also known as linearly interacting peptides (LIP).

Linear interacting peptides are usually found as connection between two protein chains. Due such importance in with respect to protein interaction, some tools have been developed to predict disordered regions. Among those tools there is FLIPPER, which predicts linear interacting regions within a protein chain starting from its structure. [17]

A few databases are currently storing information about linear interaction peptides. One of such databases is DIBS. This database contains manually curated details about linear interactions, as well as structural and functional annotations. In addition, data retrieved by running the FLIPPER predictor against the whole PDB is stored by another database, namely MobiDB 4.0. [21] [17]

1.5 Tools for searching sequence databases

Methods used for contact predictions discussed in Chapter 2 apply on multiple sequence alignments. Such alignments can be generated through various homology search and sequence alignment tools. Among the method cited in the scientific literature about contact prediction the most common ones are BLAST, PSI-BLAST and HMMER. In this section the second one is described, since it is the tool used in the pipeline presented in Chapter 2. However, it must be noticed that PSI-BLAST can be considered as an iterative evolution of the first method. Since understanding BLAST is required to understand PSI-BLAST as well, the former method is described beforehand.

1.5.1 BLAST

The Basic Local Alignment Search Tool, abbreviated BLAST, is a heuristic for searching databases and generate multiple sequence alignments. Being a heuristic, it does not retain the best solution as dynamic programming does. Instead, it sacrifices the optimal solution in order to save computational resources. This allows BLAST to retrieve good results quickly even on less powerful machines.

In BLAST, a query sequence is searched against each target sequence by means of a local similarity measure. Local similarity measures are used to align only portions of two sequences showing high conservation between each other. Instead, global similarity measures are used to favor the overall alignment, hence allowing long regions with low conservation to occur in the resulting alignment.

Taking into account two segments of the same length in both the query and the target sequences, their similarity score can be computed by summing the similarity scores of each site. The similarity score for a given site can be defined by means of a similarity matrix such as PAM120 or BLOSUM62.

The key concept in blast is the Maximal Segment Pair, abbreviated MSP. Given all the pairs of segments of the same length in the query and target sequence, the MSP is the one with the highest similarity score. Despite the length must be the same in both segments, it is bounded only by the length of the smallest sequence. Therefore an MSP is defined as locally maximal if its score can not increase by increasing its length in any direction.

Given two sequences, their MSP score can be computed in $O(n * m)$ by means of dynamic programming, with n, m the lengths of the query and target sequence respectively. As already mentioned, BLAST approximates the MSP instead of calculating the optimal solution, making it $O(n)$ or $O(m)$, depending on the longest sequence.

The approximation of the MSP between the query and any target sequence which is carried out by BLAST is based on the estimation of a similarity score S, below which it is likely to observe similarities due by chance. Then, the S score is used by BLAST

to skip sequences which are unlikely to meet such similarity score and hence to be significant matches.

Before scanning a target dataset, BLAST defines a window length w. Then, a word is defined as a segment of length w in larger sequence. Hence, a word pair is defined as two segments of the same length w, one in the query sequence and the other in the target one. Therefore, it is straightforward to assess if two sequences share a word pair whose score exceeds a threshold T. If such word is found, it is then stretched to define whether it can exceed the S threshold.

Consequently, the execution time of the BLAST algorithm is dependent from both the length of the target dataset, i.e. the number of the target sequences, and the chosen parameter T. Setting T to a low value produces a high number of hits, increasing the number of word pair to test against the threshold value S and the execution time as well. Instead, setting T too high could prevent any homologous sequence to be found.

The number of words that score at could have a similarity score above the threshold T, given a fixed query sequence, is bounded. Therefore, words which exceed that threshold are defined before scanning the dataset in BLAST. Once the list of words scoring at least T against the query sequence has been generated, scanning the whole dataset can be done in $O(nD)$, with n length of the largest target sequence and D the length of the target dataset. [5]

1.5.2 PSI-BLAST

Position-Specific Iterated BLAST, abbreviated PSI-BLAST is an extension of the BLAST method explained previously in Section 1.5.1. PSI-BLAST tries to get better results with respect to standard BLAST program both in term of computational resources required and sensitivity of its results. To do so, it implements three adjustments: it uses a new two-hit method, allows gaps to occur and involves automated profiles search.

In blast, the query sequence is compared to all the sequences in the target database. Then, those target sequences whose score is above a given threshold T are selected. Among these sequences, the ones which contain a MSP scoring at least S are returned.

The two-hit method substitutes the way in which sequences selection is made. As for BLAST, words which scores more than T against the query sequence are computed a priori. However, in PSI-BLAST a target sequence is selected only if two words exceed the threshold T within it. Moreover, the matching words must further apart than A positions. The parameter A is defined by the user as well.

For a given threshold similarity score T, the two-hits method is stricter when selecting sequences with respect to the method involved in BLAST. Then, the parameter T should be lowered in PSI-BLAST, leading to a higher number of hits. However, this does not imply that more sequences are selected, since two hits have to be found on the same target sequence.

The BLAST program could produce more than one alignment between the query and the target sequence. When multiple of such local alignments for the same target sequence are generated, the information in the resulting alignment could be biased. To avoid this issue, only significant sequences should be selected by setting a high S threshold score. Hence, PSI-BLAST considers gaps when extending the MSP region in the query and target sequences. This produces significant alignments of the target sequence against the whole query sequence, rather than small regions inside it.

PSI-BLAST borrows some ideas form methods based on profiles as well. Profiles are simple statistical models defining the probability of a given sequence of amino acids to be observed. Programs involving such methods usually require user intervention to work properly. Therefore, PSI-BLAST provides an automated way of computing profiles as Position-Specific Scoring Matrices, abbreviated PSSM.

PSI-BLAST can either use a PSSM as input as well. If a sequence is issued, the PSSM is computed during the first iteration instead. Then, a new PSSM is generated at each iteration. This enables PSI-BLAST to find homologous sequences even if they have low similarity with respect to the query sequence. [6]

1.6 Databases for biological entities

Predicting contacts between residues of a single protein chain or between those of two different interacting protein chains involves various information sources. All prediction methods exploited in chapter 2 are based on multiple sequence alignments which require a significant number of proteins whose primary structure is known, in order to work properly.

Moreover, true residues contact maps must be developed, so that prediction methods can be evaluated. To do so, the coordinates of every atom in all residues must be extracted and the euclidean distance between them can thus be computed. Once a distance matrix has been computed, applying a threshold over its values allows the definition of a contact map. A contact map holds boolean values defining whether two residues are in contact or not.

Furthermore, since various information sources are involved, each one can possibly be developed by a different curator and stored in a different format. Therefore, a mapping between all those sources is required in order to put them together and finally retrieve correct and consistent information.

In this section,a detailed description of all the information sources employed in the development of the contacts prediction and evaluation pipeline, later defined in Section 2.2 are provided. Among cited databases, the most important and well-known are the UniProt Knowledgebase for primary structures and the Protein Data Bank for secondary and tertiary structures.

1.6.1 UniProt Knowledgebase

UniProt Knowledgebase, abbreviated UniProtKB, is a large database of protein sequences. An entry of the database is made mainly by the primary structure of the protein sequence, i.e. its amino acidic sequence, and by some complementary information. Such complementary information of is referenced as annotation and provides functional information about the protein itself. Annotations can be either in human readable or machine readable format, respectively as free-text or as structured vocabularies. An example of the latter are the references to Gene Ontology and ChEBI external information sources.

Each protein sequence which can be found inside UniProt Knowledgebase is associated to a proteome, which is the set of all proteins found in a given organism. The proteome is the product of the genome by means of protein biosynthesis. A genome is the whole genetic material of a given organism, instead. Most of the entries in the UniProt Knowledgebase are derived from projects focusing on genome sequencing carried out by the International Nucleotide Sequence Database Collaboration (INSDC). INSDC is a consortium of DNA and RNA databases, which puts together resources from the DNA Data Bank of Japan (DDBJ), the European Nucleotide Archive (ENA) and GenBank.

The amount of protein sequences entering UniProtKB in the last decades saw a great increase due to large scale sequencing and the rise of metagenomics. In metagenomics, genetic material is studied directly by taking environmental samples, instead of relying on cultivation-based methods. This allowed to discover much more genetic material, yielding to discovering new proteomes as well.

Frequently, various strains of the same species are retrieved from environmental samples, leading to a redundancy issue within the database. To account for that problem, UniProtKB adopts a redundancy removal process. Such process allows to identify duplicated proteomes inside the database and to remove them. When a duplicated proteome is found, it gets discarded by moving it into the UniParc database. UniParc Archive, abbreviated UniParc, stores all protein sequences which are available in UniProtKB, without accounting for redundancy.

Moreover, a subset of non redundant proteomes held into UniProtKB composes the reference proteomes set. The goal of the reference proteomes set is to achieve a coverage over the tree of life which is as broad as possible, while limiting the number of proteomes. Proteomes in such set belong to organisms which are either very well-annotated or are of significant interest for their phylogeny and for the biomedical field.

Not only does redundancy affect proteomes, but sequences as well. To mitigate this problem, UniProt Knowledgebase provides databases containing its set of protein sequences clusterized at various similarity levels. These databases are named UniRef, followed by the percentage of similarity used to produce them. UniRef databases currently available are three, namely UniRef100, UniRef90 and UniRef50. The computation of these databases is a cascading process which starts from the highest similarity.

First of all, UniRef100 gets computed. Then, UniRef90 is computed over UniRef100. Finally, UniRef50 is computed starting from UniRef90.

Protein sequences can be stored in the UniProt Knowledgebase in one among two different partitions, namely Swiss-Prot and TrEMBL. The former is both the oldest and the smallest of such partitions, since it was established in 1986 and contains approximately half a million entries. The protein sequences stored in the Swiss-Prot database are granted to have the highest quality possible, being their annotations manually curated by hand by the personnel of the UniProt consortium. Members of the consortium which carry out the manual curation tasks are therefore called curators.

Manual curation implies the retrieval of information from scientific literature, by reading the most significant lines of text and interpreting the most important images. While this process is certainly slow and involves human manpower, on the other hand it has proven multiple times to be the most accurate method for data extraction which is currently available.

While some annotations for a protein sequence can be easily predicted through specific prediction tools or integrated from external resources, it is harder for others. Manual curation of annotations in Swiss-Prot focuses on those functional annotations which are harder to predict, since the database provides functional data used in the development and enhancement of prediction tools in bioinformatics, being it used as ground truth. Moreover, functional information about one protein sequence in Swiss-Prot is automatically transferred to similar protein sequences in TrEMBL due to transitive property. Therefore, it is important that protein sequences in Swiss-Prot and their functional annotations are kept up to date with scientific literature, as knowledge evolves through time.

The TrEMBL database has been established in 1996, ten years later with respect to Swiss-Prot. It is also the largest, since it contains around 120 million entries. It was developed to tackle the data deluge issue due to the increase of biological sequences made available by technological progress. Protein sequences in TrEMBL are automatically classified and their functional annotations are automatically generated as well. As a consequence, the human intervention is no longer required and the whole process is much less time consuming.

Automatic annotation process uses external information sources such as Interpro for the classification of protein sequences in families, sub-families and super-families. Moreover, Interpro is used for predicting both functional domains and important regions inside protein sequences. Interpro contains various predictive models of protein function and databases, such as Prosite Profiles, SMART and Pfam, to achieve these goals. Among predictive models there are both Hidden Markov Models (HMM) and profiles, which easily allow to generate domain annotations.

Furthermore, UniProt implements its own automated annotations systems. There

are two of such systems, which are complementary, namely UniRule and Statistical Automatic Annotation System (SAAS). First of all, these two systems provide annotations on the protein, e.g. the name and the functions of the protein, its catalytic activity, its interaction with other proteins, the effect of such interactions on the cell and its subcellular location. Moreover, they generate predictions regarding the protein sequence itself, e.g. the identification of active sites and positions of post-transactional modifications. [8] [23]

1.6.2 Protein Data Bank

The Protein Data Bank, abbreviated PDB is a database containing information about the observed structures of biological macromolecules. It was first established in 1971 at the Brookheaven National Laboratory, abbreviated BNL, in the Unites States. When it was born, it contained only seven structures. However, this number rapidly increased starting from 1980, mainly due to the huge amount of data produced by genomics initiative which nowadays leads the influx of data into the PDB database.

While PDB is usually referenced as a single database, it is actually composed of an heterogeneous set of resources. The most important one is the core relational database which stores data about the experiments and their outcome as coordinates in the three dimensional space of the experiment itself. Moreover, both native and derived properties of the structures are contained in a database relying on Property Object Model (POM) specifications. Instead, information derived from literature about macromolecules and crystals involved in experiments, as well as summaries, is stored in the relational Biological Macromolecule Crystallization Database (BMCD).

At its origin, PDB information could be accessed only through magnetic media. With the rise of the internet new distribution method were adopted. This allowed a broader range of users to access PDB data, coming from different fields such as biology and computer science. Nowadays, the most popular ways of retrieving data from the PDB database is through its FTP archive or web page.

New experimental observations can be inserted into the PDB database through a procedure involving three steps which can be done either by e-mail or directly through a program called AutoDep Input Tool (ADIT). First of all, structure must be submitted by the author of the experiments. Secondly, the submitted structure is inserted into the database and an unique identifier is assigned to it. Lastly, the structure goes through an iterative validation process, before being made available to users of PDB. ADIT is involved alongside Macro-molecular Exchange Input Tool (MAXIT) in the iterative validation process mentioned above. It uses the specifications for the *PDBx/mmCIF* ontology to check the correctness of submitted data.

The primary data that is contained inside the PDB database for each protein structure is the list of atoms belonging to its residue and their coordinates inside the three

dimensional space. Three dimensional coordinates can be observed by means of experiments. Nowadays, these experiments can be carried out by means of different techniques, e.g:

1. X-Ray Crystallography
2. Nuclear Magnetic Resonance (NMR) spectroscopy
3. Cryo-electron microscopy

Data is stored in *PDBx/mmCIF* format inside the PDB resource. This format is an ontology of more than 1700 terms derived from the Crystallographic Information File (CIF) format. *PDBx/mmCIF* stands for macro-molecular CIF and extends the functionalities of legacy CIF format to large molecular structures, allowing to describe complex chemistry experiments, possibly carried out through hybrid methods.

A PDBx/mmCIF formatted file stores atomic level data of biological macromolecules hierarchically, as follows:

1. Atom: it is the smallest object stored inside the file, for which coordinates have been retrieved;
2. Residue: it usually represents a proteinogenic amino acid, which contains multiple atoms;
3. Chain: is is a sequence of residues which have a peptidic bond. There could be multiple chains in a single model, many of those share the same sequence but are located in different positions inside the three dimensional space where they were observed.
4. Assembly: also called biological assembly, it describes one of the various configuration which an experimental outcome could have.

Information about a specific atom and both the residue and the chain to which it belongs are stored inside the atom record. This record is a text line, which could start either with the key word *ATOM* or *HETATM*. The former defines the entries for proteinogenic amino acids and nucleic acids, the latter identifier atoms which are found in small molecules such as water, instead. After the key, an entry is composed of six to seven value fields, as described below:

1. Sequential number of entry in file;
2. Name of the atom;
3. Name of the residue;
4. Sequential number of the residue;
5. Code of the chain;
6. Three dimensional coordinates of the atom;
7. Temperature factor.

Different experimental technique produce slightly different information. For example, X-ray crystallography is the only method which provides the temperature factor. Moreover, it provides also a resolution value, measuring the level of detail provided by the experiment carried out on a specific protein or nucleic structure. Resolution is measured in Angstrom and is said to be higher for lower values. In PDB structures with resolution lower than 1 Angstrom it is easy to identify each atom separately, while it is possible to observe only the contour of structures with more than 3 Angstrom of resolution.

Value of the resolution for PDB structures observed through NMR spectroscopy is always set to 0 Angstrom, instead. However, this technique often provide more than one macro-molecule for each experiment. Therefore, the *MODEL* key is used to index a specific molecule in a *PDBx/mmCIF* file.

Moreover, *SEQRES* key holds the list of residues which compose the polymeric molecules contained in the *PDBx/mmCIF* file, each represented by a three letters code. Not only this comprehends covalently bonded residues in each backbone of all chains and other molecules linked to such backbone as well.

Often, residues referenced in *ATOM* and the ones present in *SEQRES* do not match, due to various reasons. For example, the terminal part of the chains or loops could be missing or not observed through the experiment. Hence, residues not observed are those which are available in the *SEQRES* entry but not indexed by any *ATOM* ones.

Other keys are used for mapping a residue found inside a PDB structure to an external resource. The key *DBREF* defines an external resource itself. The key *SEQADV* express the differences that occur between the sequence stored in the PDB file and the one stored in the external resource indexed by *DBREF*. [2] [18] [20]

1.6.3 Structure Integration with Function, Taxonomy and Sequences

Structure Integration with Function, Taxonomy and Sequences, abbreviated SIFTS is a resource whose goal is to provide cross-references between sequences in the Protein Data Bank (PDB) and the UniProt Knwoledgebase (UniProtKB), as well as other resources in the bioinformatics environment, such as Pfam, InterPro and Gene Ontology (GO).

SIFTS was established in 2002 as a joint project between UniProt and PDB in Europe (PDBe) groups, both hosted by the European Bioinformatics Insititute (EBI). Moreover, all the subjects involved are part of the ELIXIR consortium. Furthermore, SIFTS was included into PDBe in 2018.

As already mentioned in the previous chapters, both the two main databases involved in the project, namely UniProtKB and PDB, saw a dramatic increase in the amount of data entering their systems. The main reasons behind such phenomenon are

the progress in technology and the arise of metagenomics. Therefore, SIFTS was developed as a mean of transferring annotations retrieved by protein structures on protein sequences and vice-versa.

Since the number of protein sequences contained in the UniProtKB is too large, most of them will never be manually annotated. Hence, UniRule is being used to obtain annotations without needing any manual intervention. As its name suggests, UniRule is a rule-based automatic annotation system trained on the set of manually annotated sequences in the UniProtKB, held inside the SwissProt partition. Then, it is straightforward that the process for automatically annotate protein sequences, held in the TrEMBL partition, can take advantage of information derived from protein structures. [4]

In SIFTS, both sequence-level and residue-level cross-references are generated between the PDB and the UniProtKB. The first one maps the sequence of a protein chain from a PDB structure to one or multiple protein sequences in UniProtKB. The second one maps a protein sequence in the UniProtKB with its three dimensional model in PDB, instead.

In the latest version of SIFTS, three different type of cross-references are available:

1. Mapping of a protein sequence from the PDB to a canonical sequence in the UniProtKB. Canonical form is the most frequent protein sequence of a set of highly similar ones, while an isoform is one of the less similar ones;
2. Mapping of a protein sequence from the PDB to all the isoform available for a canonical sequence in the UniProtKB.
3. Mapping of a protein sequence from the PDB to a cluster of the UniRef90 database.

SIFTS provides cross-reference between multiple bioinformatics resources as up-to-date as possible. It does so through a pipeline which is run weekly, at the same pace at which PDB releases new entries. Such pipeline is made of two main components, one fore each level of the cross-references: a semi-automated process for mapping protein sequences in the UniProtKB to the ones in the PDB and a fully automated process that maps residues between sequences in UniProtKB and the ones in the PDB. Moreover, a third process in involved to provide cross-references between other resources such as Pfam, InterPro or GO.

The first step of the semi-automated process for generating sequence-level mappings is performed when a new PDB structure is deposited. At that time, the name of the organism to which the entry is associated is sent to the taxonomy service of UniProt which retrieves the taxonomy identifier, also known as TaxID, according to a similarity score. Afterwards, NCBI's taxonomy database is used to retrieve the whole taxonomic lineage related to such identifier, i.e. the series of biological entities such as organisms, cells and genes on the same phylogenetic branch.

The second step is carried out a week before a PDB entry must be released. The sequence contained in the PDB entry is searched against UniProtKB through BLAST.

By taking advantage of the higher conservation of the protein structure with respect to its sequence, the whole taxonomic linkage is then assigned to the hits retrieved by querying results.

Finally, the correct entry in UniProtKB is identified through a scoring system which assigns a weight to various characteristics of the sequences retrieved from BLAST, such as the distance inside the phylogenetic tree, and the sequence similarity. Manual curation is involved to handle PDB sequences for which no hit have been retrieved through BLAST in UniProtKB and to resolve discrepancies in UniProtKB accession number originally assigned to the PDB structure.

Residue-level mapping is a process which analyzes and annotates discrepancies between sequences in the PDB and the one mapped in the UniProtKB. When a PDB sequence maps to a canonical protein sequence, then the mapping and annotation procedure is easy. However, there could be cases in which some residues in the experimental setting can not be observed, leading to gaps in the protein sequence. Such gaps, would eventually lead to errors in the output of sequence alignment algorithms.

In order to generate correct sequence alignments, gaps are considered as segment delimiters. Each segment is a sub-sequence of the original sequence whose left end and right end are referenced as amino-terminus and carboxyl-terminus, respectively. After being defined, the segments composing a sequence are searched separately against UniProtKB through BLAST and then reassembled together. Finally, annotations for amino-terminus, carboxyl-terminus, unobserved and mismatching residues are provided automatically. [3]

Cross-references are distributed per PDB entry, as XML formatted files. These files can be retrieved from the FTP service made available by PDBe. Despite being formatted in XML, the main information contained in a PDB file for each residue can be summarized as described in Table 1.1

1.7 Tools for protein contacts prediction

One of the open problems in structural bioinformatics is the prediction of intra-chain residues contacts with sufficient precision using an amino acidic sequence as input. This issue is strictly connected to the more general problem of protein folding prediction, i.e. inferring the whole tertiary structure of a protein starting from its primary structure.

Generally, methods already developed for intra-chains contact predictions take as input multiple sequence alignments. Those alignments are generated searching for a protein chain against a protein sequences database. Each method takes different aspects of input alignments into account: the simplest ones consider just the mutual information (MI) matrix and its possible regularizations and normalization. Some describe the contact matrix as a network and try to solve it through Graphical Lasso. Moreover,

Attributes		
Name	Type	Description
pdb_id	String	Identifier of the PDB structure to which current residue belongs. The file itself is named after it
pdb_chain	String	Identifier of the chain to which current residue belongs
is_observed	Bool	Boolean flag, true when the current residue has been observed in PDB structure, false otherwise
seqres_index	Int	Index of the current residue on the experimental structure
uniprot_residue_name	String	One-letter-code for the residue as mapped by the UniProtKB
residue_name	Char	One-letter-code for the residue, as mapped by the PDB
residue_name_3lett	String	Three-letters-code for the residue, as mapped by the PDB
pdb_residue_id	String	Identifier of the current residue on the PDB sequence
uniprot_index	Int	Index of the current residue on the UniProtKB sequences
uniprot_id	String	Accession number in the UniProtKB to which current residue is mapped

Table 1.1: Summary alignments table

Neural Networks have been involved in intra-chain contact predictions, taking advantage of the outcomes of the methods cited before and trying to put all their information together in order to obtain better results.

However, while it is certainly true that intra-chains contact predictions has been a quite explored issue, it is sightly less common for inter-chains contact predictions, i.e. the inference of contacts between pairs of residues in different chains. This latter issue is similar to the former one, although with some restrictions. For example, one parameter used as input for intra-chain contacts prediction and for their evaluation is their distance in term of sequence positions, especially as regards Neural Networks. These features are clearly not applicable to inter-chains contacts predictions instead.

In this chapter different methods for intra-chain contact predictions are described. These methods are also suitable for inter-chain contact predictions as well. The description takes into account the theoretical concepts which underpin contact prediction methods. The application of such notions to multiple sequence alignments is later examined in Section 1.7.

1.7.1 Covariance

Two contacting functional sites in a multiple sequence alignment are thought to coevolve. This means that when one in a pair of contacting residues changes due to evolutionary forces, the other residue changes as well. Such change is required to balance its chemical and physical differences, allowing the full sequence to maintain its native fold and therefore its functionality.

Two different sites in a multiple sequence alignment can be defined as categorical independent random variables X and Y. They can take value among the 20 standard IUPAC amino acid one letter codes. Therefore, the simplest prediction method which can be used to establish if two residues represented by variables X and Y are in contact is covariance. By computing their covariance $\mathrm{cov}(X, Y)$, their correlation coefficient $\rho_{X,Y}$ can be obtained, as well as their partial correlation coefficient $\rho_{X,Y \cdot Z}$.

Covariance between two random variables X and Y provides a quantitative measure of their joint variability. If both the variables tend to increase together, then covariance will be positive. Conversely, if one tends to decrease when also the other does, then covariance will be negative. Instead, if their behaviour is not linked then covariance will be close to zero.

Given two random variables X and Y covariance between them can be computed as the expected value of the product of their deviation from their expected values [9], i.e:

$$\mathrm{Cov}(X, Y) = \sigma_{XY} = E[(X - E[X])(Y - E[Y])] \tag{1.4}$$

By applying linearity of the expected value to covariance definition in Equation 1.4, the standard identity of the covariance can be expressed as follows:

$$\begin{aligned} \mathrm{Cov}(X,Y) &= E[(X - E[X])(Y - E[Y])] && (1.5)\\ &= E[XY - XE[Y] - E[X]Y + E[X]E[Y]] && (1.6)\\ &= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] && (1.7)\\ &= E[XY] - E[X]E[Y] && (1.8) \end{aligned}$$

In addition, if two random variables X, Y are binary the following holds, constraining their covariance within the interval $[-1, 1]$:

$$\begin{aligned} X, Y \sim Bin \in 0, 1 && (1.9)\\ E[X], E[Y] \in [0, 1] && (1.10)\\ (X - E[X]), (Y - E[Y]) \in [-1, 1] && (1.11)\\ \mathrm{Cov}(X,Y) = E[(X - E[X])(Y - E[Y])] \in [-1, 1] && (1.12) \end{aligned}$$

Once covariance has been obtained, Pearson's correlation coefficient can be computed. This allow to normalize covariance within the range $[-1, +1]$. This keeps the meaning of a positive, negative or null covariance unaltered. Given two random variables X, Y and their standard deviations σ_X, σ_Y, correlation coefficient $\rho_{X,Y}$ can be formulated as follows:

$$\rho_{X,Y} = \frac{\sigma_{X,Y}}{\sigma_X \sigma_Y} \tag{1.13}$$

Taking into account the structure of multiple sequence alignments particular importance is assumed by partial correlation coefficient. In fact, by computing covariance between any two variables X and Y as well as their correlation coefficient, the effect that another random variable Z_i has on such relationship is not considered. Instead, in the partial correlation coefficient the effect of those controlling random variables is removed, yielding a quantitative score for the relationship between variables X and Y only.

Partial correlation coefficient $\rho_{XY \cdot \mathbf{Z}}$, with $\mathbf{Z} = Z_i, ..., Z_n$ set of controlling variables and X, Y random variables, can be computed by first performing linear regressions $X \sim \mathbf{Z}$ and $Y \sim \mathbf{Z}$ against all the controlling variables. Then, residuals of both correlations e_X and e_Y must be retrieved, respectively. Finally, the correlation coefficient between them has to be calculated.

However, it might be very expensive to compute it through linear regression, particularly in high dimensional context such as the one involving multiple sequence alignments. Instead, it is better to exploit the fact that n-th order partial correlation coefficient, $|\mathbf{Z}| = n$, can be computed using just three $(n-1)$-th correlation coefficients. Therefore, by defining base case as correlation between random variables X and Y with

no controlling one, i.e. $\rho_{XY\cdot\varnothing} = \rho_{XY}$, we can rewrite partial correlation coefficient recursively, as follows:

$$\rho_{XY\cdot\mathbf{Z}} = \frac{\rho_{XY\cdot\mathbf{Z}\setminus\{Z_0\}} - \rho_{XZ_0\cdot\mathbf{Z}\setminus\{Z_0\}}\rho_{YZ_0\cdot\mathbf{Z}\setminus\{Z_0\}}}{\sqrt{1-\rho^2_{XZ}}\sqrt{1-\rho^2_{YZ}}} \quad \forall Z_0 \in \mathbf{Z} \tag{1.14}$$

This last formulation can then be computed in $O(n^3)$ by properly implementing caching, therefore avoiding computing more than one time the result for a given subset of controlling variables.

Last but not least, given a set of random variables with cardinality n, $\mathbf{X} = \{X_1, ..., X_n\}$, it is possible to compute the whole matrix of partial correlation coefficients, i.e. partial correlation coefficient between any pair of variables in set $\mathbf{X}$, in $O(n^3)$. To do so, it is required that the covariance matrix Σ is positive definite and invertible, in order to define precision matrix as its inverse, $\Theta = \Sigma^{-1}$. Therefore, matrix of partial correlation coefficients can be computed as follows:

$$\rho_{X_iX_j\cdot\mathbf{X}\setminus\{X_i,X_j\}} = -\frac{\Theta_{ij}}{\sqrt{\Theta_{ii}\Theta_{jj}}} \tag{1.15}$$

Unfortunately, observed multiple sequence alignments rarely contain all residues in every column. Thus, sample covariance matrix S is guaranteed to be singular and consequently not invertible. This makes partial correlation not directly calculable in this context. However, as will be shown later, some workaround can be applied in order to estimate partial correlation matrix.

1.7.2 Shannon's entropy and mutual information

Another common and simple tool used to infer contact maps starting from a multiple sequence alignment (MSA) is mutual information (MI). This method tries to infer contacting residues by exploiting co-evolution of those sites which are supposed to be functionally related, similarly to covariance.

Mutual information is a quantitative measure of dependence between two random variables X and Y, defined as the reduction of uncertainty about one, given that the other has been observed. However, mutual information heavily relies on the notion of Shannon's entropy and cross-entropy. Hence, their definitions are given below, before going through mutual information itself.

Shannon's entropy measures uncertainty of a random variable. By first defining a discrete random variable X, with probability mass function is $p(x)$, entropy can be defined as minus the sum of each possible value $x \in X$ times its logarithm in base b [9], i.e:

$$H_b(X) = -\sum_{x\in X} p(x)\log_b(x) \tag{1.16}$$

In the particular case the logarithm base in entropy expression is set to two, i.e. $b = 2$, entropy is said to be measured in bits.

One useful property of entropy arises from the definition of probability, for which it is known that $p(x) \in [0, 1]$, which implies $log(p(x)) \leq 0$. Therefore entropy is always non-negative, i.e:

$$H_b(X) \geq 0 \tag{1.17}$$

Another property which can be useful for the implementation of methods based on entropy is the change of logarithm base. By defining $H_a(X)$ as entropy computed for random variable X using logarithm base a, switching to base b can be done through the properties of the logarithm as follows:

$$H_b(X) = \log_b(a) H_a(X) \tag{1.18}$$

After that entropy for a single random variable X has been defined, the joint entropy between two random variables X, Y as $H(X, Y)$ has to be defined as well. By considering the joint random variable (X, Y), the two definitions are equal. Hence, the former can be considered as a special case of joint entropy, i.e. $H(X, X) = H(X)$. Formally, the joint entropy in base b of two random variables X, Y, with joint probability distribution $p(x, y)$, is defined as:

$$H_b(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_b(p(x, y)) \tag{1.19}$$

$$= -E[\log_b(p(X, Y))] \tag{1.20}$$

Moreover, it is useful to define conditional entropy of a random variable X over another random variable Y as the expected value of the conditional distributions entropies, averaged over the conditioning variable X. Hence, given X, Y random variables with joint distribution $(X, Y) \sim p(x, y)$, conditional entropy $H(Y|X)$ can be defined as:

$$H_b(Y|X) = \sum_{x \in X} p(x) H_b(Y|X = x) \tag{1.21}$$

$$= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log_b(p(y|x)) \tag{1.22}$$

$$= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_b(p(y|x)) \tag{1.23}$$

$$= -E[\log_b(p(Y|X))] \tag{1.24}$$

Relationship between joint and conditional entropy is very important in the definition of mutual information and is exploited by the chain rule theorem, i.e:

$$H_b(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(x,y)) \tag{1.25}$$

$$= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(y|x)p(x)) \tag{1.26}$$

$$= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(x)) - \sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(y|x)) \tag{1.27}$$

$$= -\sum_{x\in X} p(x)\log_b(p(x)) - \sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(y|x)) \tag{1.28}$$

$$= H_b(X) + H_b(Y|X) \tag{1.29}$$

Finally, given two random variables X and Y, mutual information between them $I(X,Y)$, i.e. the amount of information a random variable contains about the other, can be defined as follows:

$$I_b(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{1.30}$$

It can be important for the implementation of methods related to mutual information to recall the relationship which exists between entropy and mutual information itself. By means of logarithm properties, Bayes theorem, entropy and mutual information definitions, mutual information can be rewritten as the reduction in uncertainty of X given that we can observe Y, i.e:

$$I_b(X,Y) = \sum_{x\in X}\sum_{y\in Y} \log_b\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{1.31}$$

$$= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b\left(\frac{p(x|y)}{p(x)}\right) \tag{1.32}$$

$$= -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(x)) + \sum_{x\in X}\sum y \in Y p(x,y)\log_b(p(x|y)) \tag{1.33}$$

$$= -\sum_{x\in X} p(x)\log_b(p(x)) - \left(-\sum_{x\in X}\sum_{y\in Y} p(x,y)\log_b(p(x|y))\right) \tag{1.34}$$

$$= H_b(X) - H_b(X|Y) \tag{1.35}$$

By definition, mutual information is symmetric, i.e. $I_b(X,Y) = I_b(Y,X)$. This means that X contains as much information about Y as Y does about X. Then, the following equalities hold:

$$I_b(X,Y) = H_b(X) - H_b(X|Y) \tag{1.36}$$

$$= H_b(Y) - H_b(Y|X) \tag{1.37}$$

Moreover, using chain rule theorem defined above and given two random variables X, Y, it is possible to rewrite mutual information as the sum of the entropies $H(X)$, $H(Y)$ of both the variables, from which joint entropy $H(X,Y)$ is removed:

$$I_b(X,Y) = H_b(X) - H_b(X|Y) \tag{1.38}$$
$$= H_b(X) - (H_b(X,Y) - H_b(Y)) \tag{1.39}$$
$$= H_b(X) + H_b(Y) - H_b(X,Y) \tag{1.40}$$

1.7.3 Decomposition of mutual information sources

The value of mutual information computed over a multiple sequence alignment can arise from different sources. In Wollenberg and Atchley [24] four different sources of mutual information have been identified:

$$I_{\text{si}} := \text{Structural interactions} \tag{1.41}$$
$$I_{\text{fc}} := \text{Functional constraints} \tag{1.42}$$
$$I_{\text{rn}} := \text{Random noise} \tag{1.43}$$
$$I_{\text{sa}} := \text{Shared ancestry} \tag{1.44}$$

The latest two sources I_{rn}, I_{sa} represent background noise, hence they are grouped together as I_{b}. Also inter-site associations resulting from either structural interactions or functional constraints are grouped together as I_{sf}.

$$I_{\text{sf}} = I_{\text{si}} + I_{\text{fc}} := \text{Structural and functional constraints} \tag{1.45}$$
$$I_b = I_{\text{rn}} + I_{\text{sa}} := \text{Background noise} \tag{1.46}$$

Therefore, the mutual information for multiple sequence alignments can be reformulated. Taking the sum of each different factor from which it is influenced as defined above, the following formulation can be obtained:

$$I = I_{\text{si}} + I_{\text{fc}} + I_{\text{rn}} + I_{\text{sa}} \tag{1.47}$$
$$= I_{\text{sf}} + I_{\text{b}} \tag{1.48}$$

Wollenberg and Atchley [24] analyze the definition of mutual information in Equation 1.48 through artificial alignments generation. The idea behind this approach is simple: first of all, define a statistic over mutual information. Then, compute it over a real alignment and over a simulated one. Simulated alignment is parametrized to guarantee that the only source of mutual information in it is phylogeny or chance, i.e. alignment sites are independent.

Therefore, the distribution of the previously defined statistic computed on the artificial alignment can be used as a threshold value, above which it will have a specific

probability of not arising from background noise I_b. Therefore, comparing such distribution with empirical one, its probability of arising from structural or functional constraints I_{sf} can be inferred.

Method used to sample artificial alignments involves two tools. The firs is a phylogenetic tree, which can be derived by applying a neighbor-joining algorithm with mean pairwise distances to a real multiple sequence alignment. The second is an amino acid substitution algorithm. The chosen substitution algorithm is the Jones–Taylor–Thornton (JTT) in this specific case. This algorithm has the advantage of accounting for underlying phylogeny when calculating amino acid substitution probabilities. Hence, association scores computed on sequences generated through such method are guaranteed to be influenced only by background noise, i.e. by chance or due to shared ancestry.

In Martin et al. [16] correlation between background noise and number of aligned sequences is analyzed instead. Again, simulated alignments are involved, this time with independent columns, mutation probability u and two different substitution matrices: one with uniform amino acid probabilities and another with sampled ones. In both cases, observed mutual information decreases as number of aligned sequences n increases. Moreover, it can be noticed that higher mutation probability u involves also higher mutual information. This means, on the other side, that alignments with high similarity tend to have lower mutual information.

1.7.4 Mutual information normalization

Mutations should arise independently to the largest part of the residues in a sequence. Instead, most of the residues in a sequence are inherited as a block from the ancestor sequence. Therefore, even if mutations observed within a multiple sequences alignment are assumed to be independent, mutual information is greatly influenced by the phylogenetic history of the query sequence itself. [16]

To account for phylogenetic influence in contacts prediction applied to multiple sequence alignments, it is useful to understand its relation to both entropy and cross entropy computed among alignment columns. Recalling the definitions of mutual information for two random variables X, Y, the following hold:

$$I(X,Y) = H(X) - H(X|Y) \tag{1.49}$$

$$I(X,Y) = H(Y) - H(Y|X) \tag{1.50}$$

$$I(X,Y) \leq \max\{H(X), H(Y)\} \tag{1.51}$$

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{1.52}$$

Inequality 1.51 and Equality 1.52 underline the strong correlation of mutual information with either entropy and cross entropy of the columns. However, such upper bounds mean that mutual information will be higher in columns which tend to vary

often, while being low if columns vary together, but infrequently. Such behavior is not useful for contacts prediction and hence must be corrected somehow.

In Martin et al. [16], simulated alignments with correlated mutations have been exploited to better understand the problem and find a solution. In such alignments, some column pairs have been constrained to mutate together, while other columns were free to mutate independently. As expected, mutual information computed over co-evolving columns was not significantly higher than the one computed over independent ones in most cases.

Therefore, mutual information normalization methods have been empirically tested over artificially generated alignments. Dividing mutual information by cross entropy performed the best both in specificity, i.e. number of true positive values over all positively predicted ones, and sensibility, i.e. number of true negative values over the number of actually negative ones. Hence, given two discrete random variables X and Y and recalling the definition of mutual information with respect to entropy, normalized mutual information, Ir, is defined as follows:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{1.53}$$

$$I_r(X,Y) = I(X,Y)/H(X,Y) \tag{1.54}$$

Usefulness of normalized mutual information applied to multiple sequence alignments can be understood by observing its behaviour with respect to the raw score. Defining unique amino acid pairs as pairs of amino acid whose elements are not shared with any other pair, it can be noticed that mutual information score tends to zero when all possible amino acid pairs are observed in two columns of the alignment. Instead, normalized score tends to zero if all the amino acid pairs observed in two columns are unique, independently from their variability.

1.7.5 Mutual information correction for phylogeny

In Section 1.7.3, the influence of entropy in mutual information has been partially removed by applying normalization in Equation 1.54. However, mutual information computed over an alignment of homologous sequences is thought to arise as the sum of four different sources, as explained in Equation 1.47 and Equation 1.48.

Dunn, Wahl, and Gloor [10] proposes another simple method to approximate background noise, hence mutual information arising from either phylogeny or chance. This method does not rely on simulations, which can be very resource consuming when a lot of biological sequences are involved.

Such method is based on the estimation of background noise, I_b, by assuming that each position in alignment may have a particular propensity toward it, related to phylogenetic history and its entropy. Moreover, it assumes that given any pair of positions

in the alignment and their propensities to such noise, their joint propensity can be calculated as the product of the propensities of each single position.

When no signal is present in mutual information due to structural or functional constraints, hence $I_{\text{sf}} = 0$, mutual information itself is composed of noise term I_b only. Average product correction (APC) is introduced to approximate such noise term. Hence for any two sites X, Y in the alignment the following holds:

$$I(X,Y) = I_b(X,Y) + I_{\text{sf}}(X,Y) \tag{1.55}$$
$$= I_b(X,Y) \tag{1.56}$$
$$\approx APC(X,Y) \tag{1.57}$$

Furthermore, it is assumed than mutual information without the influence of structural and functional constraint between any two sites in the alignment can be approximated by the average mutual information of both those sites with respect to all other positions, divided by average mutual information computed over all positions in the set. Therefore, APC estimation can be computed between any column $X_1, ..., X_n \in X$ in multiple sequence alignment as follows:

$$APC(X_i, X_j) = I(X_i, \bar{X}) I(X_j, \bar{X}) / \bar{I}(X) \tag{1.58}$$

With $I(X_i, \bar{X})$ mean mutual information for any site X_i with respect to the others and $\bar{I}(X)$ mean mutual information computed over all sites, the following holds:

$$m = n - 1 \tag{1.59}$$
$$I(X_i, \bar{X}) = \frac{1}{m} \sum_{j \neq i} I(X_i, X_j) \tag{1.60}$$
$$\bar{I}(X) = \frac{2}{nm} \sum_{i=1}^{m} \sum_{j=i+1}^{n} I(X_i, X_j) \tag{1.61}$$

APC approximation to mutual information can be demonstrated starting by defining the product of average mutual information product of any two sites X_i, X_j with all the others, i.e:

$$I(X_i, \bar{X}) I(X_j, \bar{X}) = \left(\frac{1}{m} \sum_{k \neq i} I(X_i, X_k) \right) \left(\frac{1}{m} \sum_{k \neq j} I(X_j, X_k) \right) \tag{1.62}$$

Thus, by recalling the definition of mutual information at 1.40 the following is obtained:

$$I(X_i, \bar{X}) = \frac{1}{m} \sum_{k \neq i} I(X_i, X_k) \tag{1.63}$$

$$= \frac{1}{m} \sum_{k \neq i} (H(X_i) + H(X_k) - H(X_i, X_j)) \tag{1.64}$$

$$= \frac{1}{m} \left(mH(X_i) + \sum_{k \neq i} H(X_k) - \sum_{k \neq i} H(X_i, X_k) \right) \tag{1.65}$$

$$= H(X_i) + \frac{1}{m} \sum_{k \neq i} H(X_k) - \frac{1}{m} \sum_{k \neq i} H(X_i, X_k) \tag{1.66}$$

Then, the mean entropy of any site X_i with respect to all other sites in the alignment can be formulated:

$$H(X_i, \bar{X}) = \frac{1}{m} \sum_{k \neq i} H(X_i, X_k) \tag{1.67}$$

In addition, when the number of columns m is large the mean entropy computed on all sites but one can be approximated to the mean entropy computed on all sites, i.e:

$$\bar{H}(X) \sim \frac{1}{m} \sum_{k \neq i} H(X_k) \quad \text{for } m \text{ large} \tag{1.68}$$

By combining Equation 1.67 and Equation 1.68 in Equation 1.66 the following can be obtained:

$$I(X_i, \bar{X}) = H(X_i) + \frac{1}{m} \sum_{k \neq i} H(X_k) - \frac{1}{m} \sum_{k \neq i} H(X_i, X_k) \tag{1.69}$$

$$\approx H(X_i) + \bar{H}(X) - H(X_i, \bar{X}) \tag{1.70}$$

To successfully define APC approximation, it is of fundamental importance to assume that mean joint entropy contains an additive component, such that:

$$H(X_i, X_j) \approx H(\bar{X}) + \delta_{X_i} + \delta_{X_j} \tag{1.71}$$

Note that by definition of mean joint entropy, additive components are guaranteed to sum up to zero, hence:

$$\sum_{i=1}^{n} \delta_{X_i} = 0 \tag{1.72}$$

$$\sum_{k \neq i} \delta_{X_k} = -\delta_{X_i} \quad \text{with } i \in 1, ..., n \tag{1.73}$$

Therefore, using the definition of the average entropy for a specific site in Equation 1.67 it can be rewritten taking into consideration Equation 1.71 and Equation 1.73 as follows:

$$H(X_i, \bar{X}) = H(\bar{X}) + \delta_{X_i} - \frac{1}{m}\delta_{X_i} \tag{1.74}$$

$$\approx H(\bar{X}) + \delta_{X_i} \tag{1.75}$$

Thus, each factor in the average mutual information product from Equation 1.62 can be redefined introducing Equation 1.75 into Equation 1.70 along with a new term H_{X_i} for convenience:

$$I(X_i, \hat{X}) = H(X_i) + \bar{H} - H(X_i, \bar{X}) \tag{1.76}$$

$$= H(X_i) + \bar{H} - H(\bar{X}) - \delta_{X_i} \tag{1.77}$$

$$= (H(X_i) - \bar{H} - \delta_{X_i}) + 2\bar{H} - H(\bar{X}) \tag{1.78}$$

$$= H_{X_i} + 2\bar{H} - H(\bar{X}) \tag{1.79}$$

Hence, the Equation 1.62 for the average mutual information product can be rewritten as well, by substituting factors with the definition in Equation 1.79:

$$I(X_i, \hat{X})I(X_j, \hat{X}) = (H_{X_i} + 2\bar{H} - H(\bar{X}))(H_{X_j} + 2\bar{H} - H(\bar{X})) \tag{1.80}$$

$$= H_{X_i}(H_{X_j} + 2\bar{H} - H(\bar{X})) + (2\bar{H} - H(\bar{X}))(H_{X_j} + 2\bar{H} - H(\bar{X})) \tag{1.81}$$

$$= H_{X_i}H_{X_j} + H_{X_i}(2\bar{H} - H(\bar{X})) + H_{X_j}(2\bar{H} - H(\bar{X})) + (2\bar{H} - H(\bar{X}))^2 \tag{1.82}$$

$$= H_{X_i}H_{X_j} + (2\bar{H} - H(\bar{X}))(H_{X_i}H_{X_j} + 2\bar{H} - H(\bar{X})) \tag{1.83}$$

Considering the definition of mutual information between any two positions X_i, X_j in an alignment, it can be reformulated by introducing the approximation for the entropy form Equation 1.71 into Equation 1.40 and using terms X_{X_i}, H_{X_j} defined within Equation 1.79:

$$I(X_i, X_j) = H(X_i) + H(X_j) - H(X_i, X_j) \tag{1.84}$$

$$= H(X_i) + H(X_j) - H(\bar{X}) - \delta_{X_i} - \delta_{X_j} \tag{1.85}$$

$$= (H(X_i) - \bar{H} - \delta_{X_i}) + \bar{H} + (H(X_j) - \bar{H} - \delta_{X_i}) + \bar{H} - H(\bar{X}) \tag{1.86}$$

$$= H_{X_i} + H_{X_j} + 2\bar{H} - H(\bar{X}) \tag{1.87}$$

Mean mutual information overall $\bar{I}(X)$ can be redefined as well, by using approximation in Equation 1.71:

$$\bar{I}(X) = \frac{2}{mn}\sum_{i,j} I(X_i, X_j) \tag{1.88}$$

$$= \frac{2}{mn}\sum_{i,j} \left(H(X_i) + H(X_j) - H(X_i, X_j)\right) \tag{1.89}$$

$$= \frac{2}{mn}\left(mH(X_i) + nH(X_j) - \sum_{i,j} H(X_i, X_j)\right) \tag{1.90}$$

$$= \bar{H}(X) + \bar{H}(X) - H(\bar{X}) \tag{1.91}$$

$$= 2\bar{H}(X) - H(\bar{X}) \tag{1.92}$$

Finally, Equation 1.87 and Equation 1.92 can be used to rewrite equation 1.83:

$$I(X_i, \bar{X})I(X_j, \bar{X}) \approx H_{X_i}H_{X_j} + \bar{I}(X)I(X_i, X_j) \tag{1.93}$$

$$I(X_i, X_j) = \frac{I(X_i, \bar{X})I(X_j, \bar{X})}{\bar{I}(X)} - \frac{H_{X_i}H_{X_j}}{\bar{I}(X)} \tag{1.94}$$

The First term in right part of the Equation 1.94 represents APC correction term. Such equation approximates mutual information well when second term is close to zero. This happens for columns with entropy close to mean entropy of the alignment, since by definition $H_{X_i} = H(X_i) - \bar{H}(X) - \delta_{X_i}$.

1.7.6 PSICOV

PSICOV stands for precise structural contact prediction using sparse inverse covariance estimation. This method uses covariance to infer contacts between two residues inside the same sequence alignment. As the name suggests, it tries to be as precise as possible by tackling indirect coupling effect through regularization. Instead, methods based on mutual information described above take into account only phylogenetic bias through normalization or correction.

Sample covariance computed between all the amino acids in all the sites of a multiple sequence alignment is must be computed to define the matrix of partial correlation coefficients. The components of such matrix have already been described in Equation 1.15. Considering multiple sequence alignments, a pair of sites showing a high level of partial correlation between each other are likely to be in contact.

However, in order to obtain partial correlation coefficients it is required that the sample covariance matrix obtained from a multiple sequence alignment is invertible. This is not guaranteed in this particular setting. Instead, it is almost certainly singular since not all amino acid combinations are observed in any two sites of the alignment. This results in a number of observations which is smaller than the number of variables, thus matrix can not be inverted.

Instead of computing it directly, PSICOV estimates the sparse inverse covariance matrix. Since the percentage of contacts in a true contacts map was empirically observed to be around zero, imposing a similar target sparsity level should produce more precise results.

The sparse inverse covariance matrix can be estimated by means of graphical Lasso. Given a set of m observed random variables $x_1, ..., x_m$ with $x_i \in \mathbb{R}^n$, Graphical Lasso approximates the inverse covariance matrix by minimizing the following objective function:

$$\sum_{ij} S_{ij}\Theta_{ij} - \log(\det(\Theta_{ij})) + \rho \sum_{ij} |\Theta_{ij}| \tag{1.95}$$

where $S \in \mathbb{R}^{m,m}$ is a squared matrix whose values contain the empirical covariance computed between all the m observed random variables, recalling Equation 1.4.

Assuming that the distribution of the observed variables is a multivariate normal with mean $\mu \in \mathbb{R}^m$ and standard deviation $\Sigma \in \mathbb{R}^{(m,m)}$ and distribution $X = x_1, ..., x_m \sim \mathcal{N}(\mu, \Sigma)$, the first two terms in the objective function shown in Equation 1.95 can be defined as the negative log-likelihood function of the inverse covariance matrix Θ.

Instead, the rightmost component of the objective function is the regularization term known as ℓ_1-norm. Such regularization term allows to impose a certain sparsity coefficient to the estimated inverse covariance matrix $\widehat{\Theta}$. The sparsity coefficient of the solution can be tweaked through the coefficient $\rho \geq 0$. When ρ is set to zero, the number of non-zero values in the solution matrix $\widehat{Q}$ is maximized. Instead, when ρ increases the number of non-zero components decreases. When ρ is set to extremely high values, the third term in the objective function becomes the most important, leading to a solution $\widehat{Q}$ whose values are all set to zero.

Graphical Lasso could be slow to reach convergence. In some other cases, it could never reach convergence. PSICOV tries to solve this issue by substituting the sample covariance matrix S in Equation 1.95 with the matrix S' defined as follows:

$$S' = \lambda F + (1 - \lambda)S \tag{1.96}$$

$$\lambda \in [0, 1] \tag{1.97}$$

$$F = \text{diag}(\bar{S}, ..., \bar{S}) \tag{1.98}$$

Matrix F is a highly structured unbiased estimator and λ is the shrinkage parameter. The shrinkage parameter λ is automatically increased until S' is not singular anymore. After such matrix S' has been chosen, a solution to the dual problem can be found through block coordinate gradient descent.

As already mentioned, theoretical methods described above work when random variables considered follow a normal distribution $\mathcal{N}(\mu, \Sigma)$. The solution of the dual optimization problem can be extended to binary variables as well. When the input data is represented by a multiple sequence alignment, this requires transforming each of its

sites in a one-hot-encoded random variable with 20 levels, since there are 20 proteinogenic standard amino acids.

Therefore, for each position i, j in the alignment a sub-matrix $\Theta_{ij} \in \mathbb{R}^{(20,20)}$ is found through Graphical Lasso. Then, the sum over all amino acids combinations in positions i, j is taken as contact score S_{ij}^{contact}, i.e:

$$S_i^{\text{contact}} j = \sum_{ab} |\widehat{\Theta}_{ij}^{ab}| \tag{1.99}$$

Furthermore, PSICOV allows to apply APC correction similarly to mutual information. Hence, the final score is computed as follows:

$$PC_{ij} = S_{ij}^{\text{contact}} - \frac{S_{i\cdot}^{\text{contact}} S_{\cdot j}^{\text{contact}}}{\bar{S}^{\text{contact}}} \tag{1.100}$$

$$S_{i\cdot}^{\text{contact}} = \sum_j S_{ij}^{\text{contact}} \tag{1.101}$$

$$S_{\cdot j}^{\text{contact}} = \sum_i S_{ij}^{\text{contact}} \tag{1.102}$$

$$\bar{S}^{\text{contact}} = \sum_{ij} S_{ij}^{\text{contact}} \tag{1.103}$$

Finally, scores normalized within interval $[0, 1]$ can be obtained by applying a logistic activation function over all the residue-residue scores PC_{ij}. [13]

Chapter 2

Methods

This chapter takes into account the prediction methods involved in the final stage of the pipeline for the prediction and evaluation of contact maps. Moreover, the pipeline itself is described in detail.

For prediction methods, a general introduction is given first. Then, methods based on mutual information are introduced. Among these, there is entropy normalized mutual information, APC and ASC corrected mutual information. The last method presented is PSICOV, which does not rely on mutual information instead. Theoretical concepts underlying prediction methods are introduces as well as their implementation.

Afterwards, the whole pipeline is analyzed. Each step is described in detail on its own. Among the details provided there is the description of the input and the input. The methods involved for information storage are described as well. Moreover, each process employed in the transformation of the input data into the output information is analyzed.

2.1 Application of mutual information to multiple sequence alignments

Before describing the implementation of methods related to the mutual information, it is useful to formally define a multiple sequence alignment. In its simplest form, it can be thought as a matrix with N rows and M columns whose elements are characters. Each row in the matrix is an aligned sequence, while each its columns are referenced as alignments sites, i.e. positions within the amino acidic sequence. Its entries can assume one value among twenty characters, i.e. IUPAC one letters amino acidic codes, plus one characters which represents a gap "-". Using an alignment as a dataset, its sites are random variables and its sequences are their realizations.

However, it is well known that these variables must be one-hot-encoded, being them categorical with 20, excluding gaps. Therefore, recalling that N is the number of sequences, M the number of aligned residues and $Q = 20$ the given number of amino acidic one letter codes, a multiple sequence alignment matrix A can be formally defined

as follows:

$$A = \{0,1\}^{(N,M,Q)} \tag{2.1}$$

$$A_{i,j,k} = 1 \quad \text{if k-th residue is present in the j-th site of the i-th sequence;} \tag{2.2}$$

$$A_{i,j,k} = 0 \quad \text{otherwise.} \tag{2.3}$$

With multiple sequence alignment matrix so defined, random variables can be identified with the index of each amino acid k for each site index j. Hence, their realizations are defined as follows:

$$\{A_{i,j,k} \forall i \in N\} \in 0,1^N \tag{2.4}$$

2.1.1 Covariance

Therefore, sample covariance matrix $S \in \mathbb{R}^{(M,Q,M,Q)}$ for alignment A will hold sample covariance scores between any two amino acid indexed by k_1, k_2 in any two alignment sites indexed j_1, j_2 and is defined as:

$$S_{j_1,k_1,j_2,k_2}(A) = \frac{1}{N-1} \sum_{i=1}^{N} (A_{i,j_1,k_1} - \bar{A}_{j_1,k_1})(A_{i,j_2,k_2} - \bar{A}_{j_2,k_2}) \tag{2.5}$$

However, recalling the standard identity for covariance from Equation 1.5, i.e. $\text{cov}(X,Y) = E[XY] - E[X]E[Y]$, and assuming that the random variables are binary, i.e. $X, Y \in 0,1^N$, the expectation of a random variable is equal to the probability of a positive observation, i.e. $E[X] = p(x = 1)$. Then, it is possible to rewrite sample an entry in the sample covariance matrix S for an alignment A as follows:

$$S_{j_1,k_1,j_2,k_2}(A) = E[A_{j_1,k_1} A_{j_2,k_2}] - E[A_{j_1,k_1}]E[A_{j_2,k_2}] \tag{2.6}$$

$$= f(A_{j_1,k_1} A_{j_2,k_2}) - f(A_{j_1,k_1}) f(A_{j_1,k_1}) \tag{2.7}$$

$$= \frac{1}{N} \sum_{i=1}^{N} A_{i,j_1,k_1} A_{i,j_2,k_2} - \frac{1}{N} \sum_{i=1}^{N} Ai, j_1, k_1 \frac{1}{N} \sum_{i=1}^{N} Ai, j_2, k_2 \tag{2.8}$$

$$= \frac{1}{N} A_{j_1,k_1} A_{j_2,k_2} - \bar{A} j_1, k_1 \bar{A}_{j_2,k_2} \tag{2.9}$$

Moreover, by fixing alignment sites indices j_1, j_2, it is possible to compute a frequencies matrix $F \in [0,1]^{(Q,Q)}$ with just one cross product operation. Since binary

variables are taken into account, the following holds:

$$F_{j_1,j_2}(A) = \frac{1}{N} A_{j_1}^T \cdot A_{j_2} \in [0,1]^{(Q,Q)} \tag{2.10}$$

$$f_j(A) = \text{diag}(F_{j_1,j_2}) \in [0,1]^Q \tag{2.11}$$

$$S_{j_1,j_2} = F_{j_1,j_2}(A) - f_{j_1}(A) \otimes f_{j_2}(A) \tag{2.12}$$

$$\tag{2.13}$$

Starting from the matrix of standard deviations $D(A) \in \mathbb{R}^{(M,Q)}$ computed for each amino acid indexed by $k \in 1, ..., Q$, in each alignment site indexed by $j \in 1, ..., M$, correlation matrix $P(A) \in [-1,1]^{(M,Q,M,Q)}$ can be easily calculated as well, i.e:

$$D_{j,k}(A) = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(A_{i,j,k} - \overline{A}_{j,k} \right)^2} \tag{2.14}$$

$$P_{j_1,j_2} = S_{j_1,j_2} / D_{j_1} \otimes D_{j_2} \tag{2.15}$$

2.1.2 Shannon's entropy and mutual information

Given an alignment matrix A, the observed binary vector associated to k-th amino acid in j-th alignment site represents the realizations of a binary random variable. Recalling that the expected value of a binary random variable X is equal to its probability of being true, i.e. $E[X] = p(X = 1)$ and that the sample mean f is an unbiased estimator of the expected value, the entropy of such binary vector can be written as follows:

$$f_{j,k}(A) = \frac{1}{N} \sum_{i=1}^{N} A_{i,j,k} \tag{2.16}$$

$$H_{j,k}(A) = f_{j,k} \log f_{j,k} \tag{2.17}$$

Cross-entropy between all $k = 1, ..., Q$ amino acid indices in j-th alignment site is also called conservation. It gives information about the variability of amino acids inside the given site and tends to 1 when all residues are conserved, i.e. no amino acid changes and no gap has been observed, 0 when there are many gaps or high variability instead. This measure can be computed as the entropy of the amino acid pairs, i.e:

$$f_{j,k_1,k_2}(A) = \frac{1}{N} \sum_{i=1}^{N} Ai, j, k_1 \cdot Ai, j, k_2 \tag{2.18}$$

$$H_j(A) = -\sum_{k_1 \in Q} \sum_{k_2 \in Q} f_{j,k_1,k_2}(A) \log \left(f_{j,k_1,k_2}(A) \right) \tag{2.19}$$

Similarly, cross-entropy between any two sites in the alignment, indexed by j_1, j_2 respectively, can be computed as follows:

$$f_{j_1,j_2,k_1,k_2}(A) = \frac{1}{N}\sum_{i=1}^{N} Ai, j_1, k_1 \cdot Ai, j_2, k_2 \tag{2.20}$$

$$H_{j_1,j_2}(A) = -\sum_{k_1 \in Q}\sum_{k_2 \in Q} f_{j_1,j_2,k_1,k_2}(A)\log\left(f_{j_1,j_2,k_1,k_2}(A)\right) \tag{2.21}$$

Finally, mutual information between any alignment site j_1 and any other j_2 can be calculated by first defining joint frequencies for amino acid indices k_1 and k_2 in such sites as $f_{j_1,j_2,k_1,k_1}(A)$ and marginal frequency for amino acid index k in any site j as $f_{j,k}$. Hence, mutual information $I_{j_1,j_2}(A)$ becomes:

$$I_{j_1,j_2}(A) = -\sum_{k_1 \in Q}\sum_{k_2 \in Q} f_{j_1,j_2,k_1,k_2}(A)\log\left(\frac{f_{j_1,j_2,k_1,k_2}(A)}{f_{j_1,k_1}\cdot f_{j_2,k_2}}\right) \tag{2.22}$$

The computation of entropy and mutual operation can benefit from matrix operations on some architectures. Hence, it could be worth using them as much as possible. To do so, the dot product between two binary vectors $A_{j,k} \in \{0,1\}^N$ can be exploited, since it implicitly returns the count of joint observations:

$$f_{j_1,k_1,j_2,k_2}(A) = \frac{1}{N}\sum_{i=1}^{N} Ai, j_1, k_1 \cdot Ai, j_2, k_2 \tag{2.23}$$

$$= \frac{1}{N}\left(A^T_{j_1,k_1} \cdot A_{j_2,k_2}\right) \tag{2.24}$$

By applying the same strategy to binary matrix at the j-th alignment site, i.e. $A_j \in \{0,1\}^{(N,Q)}$, one can easily compute joint frequencies among all amino acid pairs as a matrix $F_{j_1,j_2}(A) \in [0,1]^{(Q,Q)}$ whose components indexed by k_1 and k_2 are defined as follows:

$$F_{j_1,j_2;k_1,k_2}(A) = \frac{1}{N}\sum_{i=1}^{N} Ai, j_1, k_1 \cdot Ai, j_2, k_2 \tag{2.25}$$

$$F_{j_1,j_2}(A) = \frac{1}{N}A^T_{j_1} \cdot A_{j_2} \tag{2.26}$$

Therefore, it is simple to implement cross entropy matrix for any two sites j_1 and j_2 in alignment A as matrix $H(A) \in [0,1]^{(M,M)}$, whose elements are defined as follows:

$$H_{j_1,j_2}(A) = -\sum_{k_1 \in Q}\sum_{k_2 \in Q} F_{j_1,j_2;k_1,k_2}\log(F_{j_1,j_2;k_1,k_2}) \tag{2.27}$$

Mutual information matrix $I(A) \in \mathbb{R}^{(M,M)}$, whose elements are indexed by two sites j_1, j_2 in the alignments, contains mutual information computed between those two

sites. Recalling the definition of entropy computed between two sites of the alignment $H_{j_1,j_2}(A)$, and noticing that marginal entropy is equal to cross entropy between one alignment site and itself, i.e. $H_j(A) = H_{j_1=j_2=j}(A)$, the elements of the mutual information matrix can be computed as follows:

$$I_{j_1,j_2}(A) = H_{j_1}(A) + H_{j_2}(A) - H_{j_1,j_2}(A) \tag{2.28}$$

2.1.3 Pseudocount for low number of sequences

In Buslje et al. [7] a pseudocount parameter λ is introduced in the formula for both marginal and joint frequencies. This parameter corrects for low number of observation. Given two columns indexes j_1, j_2 and two amino acid indexes k_1, k_2 in the input alignment A, their marginal frequencies can be computed as follows:

$$F_{j_1,j_2;k_1,k_2} = \frac{\lambda + \sum_{i=1}^{N} A_{i,j_1,k_1} \cdot A_{i,j_2,k_2}}{\sum_{k1,k2} \lambda \sum_{i=1}^{N} A_{i,j_1,k_1} \cdot A_{i,j_2,k_2}} \tag{2.29}$$

Correction for low number of observation can be introduced in marginal frequencies similarly to joint ones. Given a column index j_1 and any amino acid index k_1 in the input alignment A, it becomes:

$$f_{j_1;k_1} = \frac{\lambda + \sum_{i=1}^{N} A_{i,j_1,k_1}}{\sum_{k} \lambda \sum_{i=1}^{N} A_{i,j_1,k}} \tag{2.30}$$

The pseudocount λ has limited influence on frequency when a large number of aligned sequences are present in the alignment. Conversely, it plays a key role in columns with a small number of observed residues and a high number of gaps. Optimal pseudocount value for contacts prediction has been found to be $\lambda = 0.05$.

2.1.4 Entropy normalized mutual information

Normalized mutual information matrix $I_b(A) \in \mathbb{R}^{(M,M)}$ can be computed dividing mutual information matrix I_{j_1,j_2} by cross entropy. Therefore, its elements are defined as follows:

$$I_{b_{j_1,j_2}}(A) = \frac{I_{j_1,j_2}(A)}{H_{j_1,j_2}(A)} \tag{2.31}$$

$$= \frac{H_{j_1}(A)}{H_{j_1,j_2}(A)} + \frac{H_{j_2}(A)}{H_{j_1,j_2}(A)} - 1 \tag{2.32}$$

2.1.5 Phylogeny corrected mutual information

Once mutual information matrix $I(A)$ has been obtained, it is easy to compute APC correction defined in Equation 1.58 as well. First, an array for mean mutual information between each site in alignment A and all the others except self, $I_{\bar{j}} \in \mathbb{R}^M$, has to be

defined. Hence, its elements $I_{\bar{j},j_1}(A)$ can be computed as follows:

$$I_{\bar{j},j_1}(A) = \frac{1}{n-1} \sum_{j_2 \neq j_1} I_{j_1,j_2} \tag{2.33}$$

Moreover, the mean mutual information overall must be defined for alignment A, $\bar{I}(A) \in \mathbb{R}$, as the mean values of non diagonal entries in mutual information matrix, i.e:

$$\bar{I} = \frac{1}{n(n-1)} \sum_{j_1=1}^{n} \sum_{j_2 \neq j_1} I_{j_1,j_2}(A) \tag{2.34}$$

Therefore, by recalling Equation 1.58, APC regularized mutual information matrix for alignment A, $I_r(A) \in \mathbb{R}^{(M,M)}$, can be computed by subtracting from mutual information between two positions indexed by j_1, j_2 respectively, the multiplication of their mean mutual information divided by overall mean mutual information. Hence, by noticing that the subtracted term is $APC(A)$ regularization term, the following is obtained:

$$I_r(A) = I(A) - \frac{I_{\bar{j}}(A) \otimes I_{\bar{j}(A)}}{\bar{I}(A)} \tag{2.35}$$

$$= I(A) - \mathrm{APC}(A) \tag{2.36}$$

Furthermore, background noise can be estimated through an additive model, instead of the multiplicative one used by APC. Such additive model is called Average Sum Correction (ASC). ASC corrected mutual information I_p is defined as follows:

$$I_p(A) = I(A) - \left[I_{\bar{j}}(A) \oplus I_{\bar{j}}(A) - \bar{I}(A) \right] \tag{2.37}$$

$$= I(A) - ASC(A) \tag{2.38}$$

2.2 Pipeline

In this section, the pipeline built to generate and analyze contact predictions over linearly interacting peptides (LIPs) is described. Each step is associated to a specific script, which has been developed in python. The scripts were developed to be run in the same order in which they are presented below. The description covers each different step separately, describing its input and the output that it produced.

The first step in the pipeline the dataset initialization. This step takes as input a file containing predictions for linear interacting peptides over protein chains in PDB. Then, it initializes the folder structure for the whole input dataset. The folder structure is generated in a way that allows computations on each different PDB structure to be executed separately. This allows to execute the subsequent steps in parallel, exploiting the computational capacity of a cluster at its best.

The second step is intended to be executed on a single PDB structure or a batch of such structures, as well as the next steps. The goal of this step is to define a mapping between each chain in the protein structures, the LIP predictions and the UniProt entries. Moreover, in this step the three dimensional coordinates of each all the atoms in each residue are retrieved.

The third step uses the atomic coordinates retrieved in the previous step to generate two different distance matrices: one between heavy atoms and one between carbon atoms only. After the distance matrices have been generated, a threshold value is applied on them to obtain the contact maps.

In the fourth step, multiple sequence alignments are generated. Differently form the other points in the pipeline, this one involves two scripts. The first one executes PSI-BLAST on all the UniProt sequences obtained in the second step, producing their multiple sequence alignments. The following one maps back the columns in the generated alignments to protein chains. The latter is responsible to produce joint sequence alignments as well.

The fifth step runs prediction methods on both intra-chain and inter-chain multiple sequence alignments to obtain contact predictions. Although just one script has been developed for that purpose, it can be set up to predict contacts either through mutual information based method or by means of PSICOV.

The last step involves the generation of various summary tables. The associated script takes as input the folder initialized in the first step. It is not meant to be executed in parallel with any other script, therefore it scans the whole dataset sequentially. The final analysis has been carried out through interactive Jupyter Notebooks, starting from the tables generated in this step. The results of such analysis can be found in Chapter 3.

2.2.1 Initialization of output dataset

The dataset initialization phase is carried out by the `init_db.py` script. It takes as input a file containing residue level predictions of linear interacting peptides made on different protein chains in PDB. Hence, for each of such predictions it contains the reference to the chain in the PDB database and its amino acidic sequence as well.

The format of the input file is similar to the FASTA one. A FASTA formatted file contains one or more protein sequences. Each protein sequence is expressed as an ordered string where each residue is expressed with the one letter code of its amino acid. In FASTA, each sequence can lay on one or more consecutive lines in the file. Moreover, each sequence is preceded by a description line. A description line starts with the character >, followed by free text. Usually, such text is formatted according to the database where the FASTA file is stored. The description line is often referenced as FASTA header and works also as a separator between different sequences.

```
1  >1a6b_B
2  GERRRSQLDRDQCAYCKEKGHWAKDCPKKPRGPRGPRPQT
3  00000001111111111111111111110000000000000
4  >1ar7_4
5  GAQVSSQKVGAHESTINYTTINYYRDSASNAASKQDFSQDPSKFTEPIKDV
6  11111100000000001111111111111111111111111111111111
7  >1a93_B
8  ---MRRKNHTHQQDIDDLKRQNALLEQQVRAL
9  00111111111111111111111111111111
10 ...
```

Listing 2.1: Few lines in an input file example.

Listings 2.1 shows two entries in an example of input file. Differently from a common FASTA file described above, an entry does not end with its amino acidic sequence. Instead, such sequence of one letter codes is followed by another alphanumeric sequence of the same length, representing boolean predictions. Each value in the latter sequence is 1 if the residues in that position has been predicted as linearly interactive, 0 otherwise. Moreover, the third entry shows some gaps in the sequence of residues at line 8. This could happen because of a mismatch in the mapping between LIP predictions and and PDB sequences, produced by the LIP predictor itself.

The script `init_db.py` requires two parameters to be set by the user. The first is the path to an output directory, where the whole dataset will be generated and stored. The other is the path to an input file formatted as shown in Listings 2.1. These two parameters are hereby defined as `/out/dir` and `/in/path`.

The script generates the output directory if it is not already existent. Then, it goes through each entry in the input file sequentially, i.e. taking into account one entry at a time. Each entry has a header which starts with the > character, as for FASTA files. Then, the header contains both the identifier of the protein structure in within the PDB dataset and the identifier of the protein chain upon which predictions were made. These two identifier are separated by an underscore character. Hence, the script defines these identifiers separately by means of regular expressions. Taking the first entry in Listings 2.1 as example, its protein structure identifier is `1a6b`, while the protein chain identifier is `B`. Both have been retrieved from the header of the entry in the first line.

For each scanned entry, its protein structure and chain identifier in the PDB database are defined as `pdb_id` and `chain_id` respectively. A sub-folder is then created within the output directory and named after the PDB structure identifier, if not already existent. In the example above, the folder associated to the first entry would be `/out/dir/pdb_1a6b`.

Then, the script copies the whole entry in a file named after the protein chain

identifier. The file is placed in a folder within the one created for the protein structure. The folder contains other files as well, each one containing input entries for different chains in the structure taken into account. These files are named after their chain identifier. Recalling the same example as before, the path to the file would be `/out/path/pdb_1a6b/lips/B.fasta`.

Other than the one containing LIP predictions, another folder is created within the folder associated to the protein structure. That folder is empty and named `/aligns`. As the name suggests, this folder is used to store both intra-chain and inter-chain alignments in the following steps of the pipeline. Taking into account the example entry, the path to the alignments sub-folder would be `/out/dir/pdb_1a6b/aligns/`.

2.2.2 Linking residue-level information between external resources

The dataset building step retrieves information about PDB structures identified in the previous one. It retrieves and stores the residue level mapping between every chain in these structures and both the entries in the UniProt database and the LIP predictions. To do so, it exploits the SIFTS dataset, already mentioned in Section 1.6.3. Moreover, given a protein chain, it retrieves the coordinates of its atoms by looking into PDB files.

This step is implemented by the script `make_db.py`. The script takes as input the path to a folder associated to a protein structure initialized by the script `init_db.py`, or a batch of them. Each of these folders are taken into account sequentially within the same batch. Nevertheless, the script can be executed on multiple batches either on the same or different machines without any conflict.

The information retrieved by the script `make_db.py` for each input protein structure is stored in a HDF5 file. Given a specific protein structure, the file is named `.h5` and is created within the folder of the structure initialized by the script `init_db.py`. For example, the information for the PDB structure `1a6b` would be stored at `/out/dir/pdb_1a6b/.h5`.

HDF5 stands for Hierarchical Data Format 5. As the name suggests, it is a hierarchical format. Hence, similarly to the JSON format it allows to retrieve data quickly. Differently from the former, POSIX-like syntax can be used to retrieve specific values. Moreover, HDF5 allows to easily store large arrays and multi-dimensional matrices. Moreover, it natively handles data compression, saving up some disk space. Other than these useful features provided under the hood by HDF5, a file in this format can be though as a common dictionary.

Each path given as input to the script `make_db.py` points to a folder, which is in turn associated to a specific PDB structure. Since the folders were initialized by the script `init_db.py`, it is possible to extract the identifier of the PDB structure directly from its path. Given the PDB identifier of a structure, the information about its chains is retrieved from SIFTS files. These files are available form either the SIFTS website or FTP service in XML format. However, since many SIFTS files were already available in

MJSON format, these were used instead. The path to the folder containing SIFTS files has to be user defined.

Given a PDB structure, the script `make_db.py` looks into the related SIFTS file to define its protein chains and their identifiers. Then, it goes through a protein chain at a time. For each chain contained in a PDB structure, SIFTS provides the mapping between its residues and other resources. Often, just one UniProt sequence can be mapped on a PDB chain. However, it might happen that two different UniProt sequences map on the same PDB chain. In the latter case, only the UniProt sequence which has the best coverage over the PDB chain was kept, while the others were discarded.

Once the associated UniProt entry has been defined for a given protein chain, the scrips generates a FASTA file within an alignments folder defined by the user. The FASTA file is named after the UniProt entry which maps the protein chain. Since only one alignment is generated for each UniProt entry, all the times the `make_db.py` script is executed, it should point to the same alignments folder.

Moreover, for each residue in a PDB chain many atoms can be observed, each with its own three dimensional coordinates. Atom coordinates are stored within `PDBx/MMCIF` files, usually provided through the FTP service of wwPDB. The script `make_db.py` allows to define a folder where those files can be found for each PDB structure and to automatically download them if they were not available.

The script initializes a list of 100 atoms for each residue in a protein chain. The initial values of these coordinates are all set to infinite. Doing so, residues for which no atoms were found are guaranteed to produce an infinite distance and any false contact. Then, the coordinates for the first and second atoms are filled with the coordinates of carbon-alpha and carbon-beta atoms, respectively. The coordinates for the other atoms are stored in the following positions, instead. Since not all the positions in the initial list will probably be filled, the ones which are found to be empty for all the residues are trimmed out, with exception of the first two.

Then, the script maps the predictions of the linear interacting peptides over each PDB chain. Given a protein structure and a chain within it, the script `init_db.py` generated a file of LIP predictions for that chain only. Taking the chain `B` of the PDB structure `1a6b` as an example, its LIP predictions file would be stored at `/out/path/pdb_1a6b/lips/B.fasta`. Some predictions could be mismatching with respect to the residues sequence defined in SIFTS. In this case, no residue is defined to be linearly interacting.

Table 2.2 and Table 2.3 show the first residue level mappings stored for the protein chain `A` belonging to the PDB structure `3m91`. It can be noticed that the first six entries do not have a valid mapping from UniProt to PDB, therefore their atom coordinates are all set to infinite. The first three residues with matching correctly between PDB and

UniProt are not predicted to be linear interacting. Instead, the residues number nine and ten have been predicted as LIP.

		Chains			
		A	B	C	D
Chains	A	A	A	AC	AC
	B	A	A	AC	AC
	C	CA	CA	C	C
	D	CA	CA	C	C

Table 2.1: Matrix defining the identifiers of both inter-chain and intra-chain alignments between reference chains in the PDB structure `1cxp`.

The script defines the reference chains within each protein structure as well. Given a group of protein chains which share the same protein sequence, the reference chain is one chain used to identify the entire group. In this case, it is the protein which comes first in alphanumeric order. Afterwards, the joint alignments between reference chains are defined by concatenating their identifiers.

Reference chains are stored in matrices, such as the one shown in Table 2.1 for the PDB structure `1cxp`. The structure contains four different chains: `A`, `B`, `C`, `D`. Since chains `A` and `B` share the same amino acidic sequence, the former has been defined as reference. The same happens between protein chains `C` and `D`. It can be observed that join alignments are made only between reference chains. This allows to reduce redundancy in the generation on multiple sequence alignments.

2.2.3 Distance matrices calculation

To evaluate contact prediction methods, true contact maps have to be developed. Contact maps can be obtained by applying a threshold over distance matrices. Distance matrices contain euclidean distances between residues in two protein chains. These matrices are guaranteed to be squared and symmetric when a protein chain is compared with itself.

Three dimensional coordinates are not available at residue level. Instead, they are available at atomic level. Euclidean distance between any two atoms A_1, A_2 can be computed as follows:

$$d_{\text{atom}}(A_1, A_2) = \sqrt{(A_{1,1} - A_{2,1})^2 + (A_{1,2} - A_{2,2})^2 + (A_{3,1} - A_{3,3})^2} \tag{2.39}$$

$$= \sqrt{\sum_{i=1}^{3} (A_{1,i} - A_{2,i})^2} \tag{2.40}$$

Each residue contains many atoms. Therefore, the distance between two residues must be defined as a function of the distances of their atoms. Any two residues R_1, R_2

Attributes								
seqres_ix	pdb_ix	uniprot_ix	is_aligned	is_observed	is_amino	is_standard	is_lip	resnames
1	-	46	False	False	False	False	False	S
2	-	47	False	False	False	False	False	H
3	-	48	False	False	False	False	False	A
4	-	49	False	False	False	False	False	P
5	-	50	False	False	False	False	False	T
6	-	51	False	False	False	False	False	R
7	52	52	True	True	True	True	False	S
8	53	53	True	True	True	True	False	A
9	54	54	True	True	True	True	True	R
10	55	55	True	True	True	True	True	D
...	...	...	...	...	...	...	...	...

Table 2.2: First 10 residues mapped between PDB, UniProt and LIP predictions for chain `A` in PDB structure `3m91`

			Atom coordinates									
Attributes			1st atom (C_α)			2nd atom (C_β)			3rd atom			...
seqres_ix	pdb_ix	uniprot_ix	x	y	z	x	y	z	x	y	z	...
1	-	46	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	...
2	-	47	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	...
3	-	48	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	...
4	-	49	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	...
5	-	50	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	...
6	-	51	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	...
7	52	52	-7.6	-784.4	-56.0	-8.4	-784.8	-57.2	-7.4	-786.0	-54.2	...
8	53	53	-5.3	-787.5	-55.2	-4.3	-788.2	-56.2	-4.6	-787.5	-52.9	...
9	54	54	-3.2	-785.1	-53.1	-2.3	-784.0	-53.6	-0.2	-782.8	-53.0	...
10	55	55	-6.3	-783.6	-51.5	-7.5	-782.8	-52.3	-6.9	-784.5	-49.4	...
...	...	...	...	...	...	...	...	...	...	...	...	...

Table 2.3: Atom coordinates for the first 10 residues mapped between PDB and UniProt for chain `A` in PDB structure `3m91`

can be represented by two vectors containing N, M atoms, respectively. Each item in these two vectors is a three dimensional coordinate, i.e. $R_1 \in \mathbb{R}^{N,3}$ and $R_2 \in \mathbb{R}^{M,3}$. Hence, it is possible to define the distance between R_1, R_2 as the minimum euclidean distance between their atoms, i.e:

$$d_{\text{residue}}(R_1, R_2) = \min \left\{ d_{\text{atom}}(A_i, A_j) \forall_{j \in 1,..,M} \forall_{i \in 1,..,N} \right\} \tag{2.41}$$

In the pipeline, distance matrices are computed by the `make_dmat.py` script. The script takes as input a batch of directories, each associated to a different PDB structure. Each directory must contain the HDF5 file generated beforehand by the script `make_db.py`. Then, `make_dmat.py` takes advantage of the atomic coordinates stored inside such file. According to scientific literature, two distance matrices have been computed between each pair of chains. On each of such distance matrices, a different threshold has been applied to obtain a specific contact map.

The carbon atom distance matrix takes into account only carbon-β and carbon-α atoms for each residue. It defines the distance between two residues as the distance between their carbon-β atoms. Instead, if coordinates for both carbon-β atoms are not available, then the distance between carbon-α atoms is considered. The first two positions in the array of atomic coordinates are always assigned to the carbon-α and carbon-β atoms respectively, as described in Section 2.2.2. Therefore, retrieving their values is straightforward.

Heavy atoms in a residue are defined as the set of all atoms excluding hydrogen ones. In an heavy atoms distance matrix only heavy atoms are considered. Hence, given any pair residues within such matrix, its value is computed as the minimum euclidean distance between all their heavy atoms. Atomic coordinates are retrieved only for heavy atoms by the script `make_db.py`. Thus three dimensional coordinates needed to compute heavy atoms distance matrices are easily retrievable.

In order to turn distance matrices in contact maps, it is required to apply a threshold on them. Distances contained in distance matrices are expressed in Angstrom (Å). Two residues are considered to be in contact in carbon atoms distance matrices if they are closer than 8Å. Instead, this threshold value lowers to 6Å in case of heavy atoms distance matrices.

Once computed, both distance matrices and contact maps are stored in the same HDF5 file defined by the `make_db.py` script. Note that different chains within the same protein structure can share the same sequence but not the same position in space. Then distance matrices are computed between all possible pairs of chains, ignoring whether they are reference chains or not. Later, the contact map for a reference chain can be obtained as the sum of all contact maps referencing to it.

		x	−6.3	−7.5	−6.9	...
		y	−783.6	−782.8	−784.5	...
x	y	z	−51.5	−52.3	−49.4	...
−7.6	−784.4	−56.0	4.75	4.03	6.64	...
−8.4	−784.8	−57.2	6.19	5.37	7.95	...
−7.4	−786.0	−54.2	3.78	3.72	5.05	...
...	...	...	...	...	...	...

Table 2.4: Atomic distances between the 7th and 10th residue in chain `A` of the PDB structure `3m91`

2.2.4 Multiple sequence alignments generation

Generating the multiple sequence alignments is the most cumbersome step of the pipeline. Unlike the other ones, this step requires multiple scripts to be executed in the same order as they are presented here. The first script fetches query sequences from UniProt. The second script is required to search each of those query sequences against UniProt to find homologous sequences and generate multiple sequence alignments. The latest one maps those multiple sequence alignments back to protein chains.

The first script, namely `/bash/fetch.sh`, calls an utility which automatically fetches the required protein sequences from an instance of the UniProt database already available on the cluster where the pipeline has been executed. The second script, namely `run_blast.py`, is executed for each of the fetched UniProt entries. It searches an UniProt entry against the whole UniProt dataset. It uses the PSI-BLAST algorithm already mentioned in Section 1.5.2. The result obtained through these two step is a list of FASTA files, each containing the pairwise alignment of a query sequence against the significant one found in UniProt. All those files are stored in the same folder, which is hereby defined as `/out/dir/aligns/`.

Once the alignments for the UniProt sequences have been obtained and are available at `/out/dir/aligns/`, the alignment for each protein chain has to be generated. This task is performed by the script `parse_aln.py`. Similarly to the ones performing the previous steps, it takes as input a batch of folders associated to different PDB structures initialized beforehand using `init_db.py` and `make_db.py`.

Given a protein structure in the input batch, the script `parse_aln.py` retrieves the residue level mappings between its reference chains and the UniProt sequences defined in the previous steps. Then, it uses these mappings to generate a new alignment for each chain selecting some columns form the alignment of the associated UniProt sequence. The columns selected are those which map on the PDB chain. After the alignment for the PDB chain has been made, the rows with an occupancy score lower than 50% are trimmed out.

Once the alignments for the reference chains of protein structure have been done, joint alignments between them must be developed. In an ideal case, a joint alignment would concatenate any pair of chains from the input alignments which are known to be interacting. Unfortunately, this information is not available. Therefore, the interaction between two sequences in the input alignments has been approximated by the fact that they belong to the same organisms. Hence, the taxonomy identifier of each sequence has been chosen as the join feature, since it is available for each UniProt entry. As for intra-chain alignments, sequences with occupancy score lower than 50% have been removed.

It is possible that more than one sequence within the alignment of a reference chain is associated to the same taxonomy id. Although it is improbable, given two alignments whose sequences belong to the same organism would produce an enormous alignment with a number of sequences equal to the product of their sequences. Then, the number of sequences have been limited to one for each taxonomy identifier in the alignment. Hence, for a given taxonomy identifier, only the sequence with the highest E-value has been kept within the input alignments. Since the sequences in the alignments are ordered from lower to higher E-value, this task is straightforward. Moreover, it guarantees that the resulting alignments contain at most the same number of sequences of the input ones.

Furthermore, it could be useful to reduce the number of sequences contained in the alignments to reduce the computational burden of the contact prediction methods as well, while keeping the initial information unaltered. Hence, only one sequence is kept for any group of sequences which share an identity score higher than a fixed threshold value. This optimization step has been implemented through the Hobohm-1 clustering algorithm described in Section 2.2.5. The identity score threshold has been fixed to 60%.

After being clusterized, both intra-chain and inter-chain alignments are stored in FASTA formatted files. Since each alignment is generated for one or two joint chains in the same protein structure, it will be stored in the folder associated to the protein structure itself. Taking as example the PDB structure `1a6b`, the alignments made for its chain `A` would be stored at `/out/dir/pdb_1a6b/aligns/A.fasta`. All the other alignments would be stored in the same folder, as well.

2.2.5 Hobohm clustering algorithm

Sequence clustering algorithms allow to define a few representative sequences among many others, according to a sequence similarity score. A cluster is a partition of the initial set of sequences, identified by the representative sequence itself. In this section, two iterative sequence clustering algorithms are presented, namely Hobohm-1 and Hobohm-2. While being essentially the same algorithm, they differ in the definition of representative sequences. [12]

The Hobohm-1 clustering algorithm takes as input a sorted list of protein sequences. It iterates until all the sequences within the input list have been assigned to a cluster. At each iteration, the algorithm initializes a new empty cluster. The representative sequence of such cluster is the first sequence in the list. Then, all the other sequences are compared with the representative one, by means of a similarity score. If the similarity score exceeds a given threshold, the sequence is assigned to the current cluster. Then, sequences belonging to such cluster are removed from the list and a new iteration is done.

As already mentioned, the Hobohm-2 sequence clustering algorithm is slightly different to the Hobohm-1 algorithm described above. Given a list of sequences as input, it does not select the first one as reference sequence at each iteration. Instead, it chooses as representative the sequence which has the highest number of neighbors. A neighbor is another sequence in the list which exceeds the similarity score threshold defined beforehand. Then, the list of neighbors for each sequence in the list must be computed initially and kept up to date at each iteration. This increases the computational burden with respect to the first version of the algorithm.

						Cluster	Score
F	R	P	N	K	V	1	1.00
F	R	-	N	K	V	1	1.00
F	A	S	N	K	V	1	0.66
F	A	-	-	-	V	1	0.66
A	B	E	N	K	A	5	1.00
-	R	P	N	K	-	1	1.00
A	B	S	N	K	-	5	1.00

Table 2.5: Example of Hobohm-1 algorithm result.

Hobohm-1 has been implemented within the pipeline in Section 2.2.4. In this particular setting, it takes as input multiple sequence alignments. The chosen sequence similarity function was percent identity. Given a pair of aligned sequences, percent identity is the fraction of aligned positions containing the exact same residue, excluding those positions with gaps.

Table 2.5 shows an example multiple sequence alignment with 6 columns and 7 rows. Clusters have been computed using percent identity as similarity score and setting the threshold value to 0.60. Each resulting cluster is defined by the index of its representative sequence. Moreover, the score columns contains percent identity of any sequence with respect to the representative one. There are two clusters in the table, one represented by the first sequence and one by the fifth.

Table 2.6 presents in detail how the percent identity between the first and the fourth sequences in the example above has been computed. The identity score does

	F	R	P	N	K	V	
	F	A	-	-	-	V	Sum
Equal	1	0	0	0	0	1	2
Gap	1	1	0	0	0	1	3

Table 2.6: Example of percent identity computation.

not take into account the positions where there are gaps. Hence, positions from third to fifth are excluded. Instead, in two of the three non-gapped positions there are pairs of equal residues. Therefore the resulting identity score is $2/3 = 0.66$. Since it is higher than the threshold value 0.60, the second sequence has been assigned to the first cluster.

2.2.6 Prediction of intra-chain and inter-chain contact maps

Contact predictions step applies method introduced in Section 1.7 on multiple sequence alignments generated in Section 2.2.4 to estimate contact maps. While predictions can be obtained either PSICOV or methods based on mutual information, all the computations are carried out through the script `pred_cmap.py`.

The script `pred_cmap.py` takes as input one or more folders associated to a PDB structure, similarly to the majority of the other scripts in the pipeline. The script goes through each folder sequentially. Hence, predictions are made for a single PDB structure at a time.

Given a protein structure, a contact map must be estimated for all the possible chain pairs, whether an input alignment is available. Moreover, it must be estimated for single chains as well. The `pred_cmap.py` script takes advantage of the information defined previously by the `make_db.py` to avoid redundant estimations. Protein chains having the identical sequence are grouped by reference chain. Hence, a contact map is computed if the input alignment involves only reference chains.

Predicted contact maps are stored in HDF5 formatted files, different from the one defined in `make_db.py`. A different file is created for a different prediction method. Considering the example PDB structure `1a6b`, the path to the HDF5 file containing PSICOV predictions would be `/out/dir/pdb_1a6b/.psi.h5`, while the one for mutual information results would be `/out/dir/pdb_1a6b/.mi.h5`.

Inside the resulting HDF5 file, an estimated contact map is indexed by the identifier of its chain. If the contact map involves two different chains, it is indexed by the concatenation of their identifiers. The script stores estimated contact maps involving only reference chains first. In case of inter-chain contact maps, the transposed one is stored as well and indexed by the inverted concatenation of the chains identifiers. For predictions involving non-reference chains, a pointer to the associated contact map is generated. This allows to save either storage space and computational resources since no prediction is made on the same alignment more than one time.

Recalling the example showing reference chains in the PDB structure `1a6b` in Table 2.1, predictions would be generated for alignments of chains `A`, `C` and for their joint alignment indexed by `AC`. Then, predicted matrix for joint alignment `CA` would be made by transposing the former one. Finally, for alignments of the other chains a reference to previously predicted contact maps is made. In the case of chain `B`, a reference to the results obtained for chain `A` would be made.

2.2.7 Generation of summary tables

All the previous steps in the pipeline store most of the information in HDF5 files, as well as FASTA files for alignments. Storing information is such a sparse way within the folder initialized in Section 2.2.1 allows multiple parallel execution of the same script in the same step of the pipeline. On the other hand, retrieving overall information about the dataset is not an immediate task.

To overcome this issue, the script `scan_db.py` has been developed. As the name suggests, it scans the whole dataset and looks into both HDF5 formatted files and multiple sequence alignments. Then, it retrieves three tables containing statistics for protein chains, alignments and alignments between reference chains, respectively. Finally, those table are made available as tab separated files.

Attributes		
Name	Type	Description
pdb_id	String	ID of the PDB structure
pdb_resol	Float	Resolution of the experiment, zero if *Nuclear Magnetic Resonance (NMR)* was used
chain_id	String	ID of the protein chain
align_id	String	ID of the reference chain
uniprot_id	String	ID of the UniProt entry which best maps the PDB structure
crystal_len	Int	Number of residues in the crystal used during the experiment
num_aligned	Int	Number of residues mapped by either PDB and UniProt on crystal
num_lips	Int	Number of LIP predictions mapped by either PDB and UniProt on crystal

Table 2.7: Summary chains table

Table 2.7 contains information for every protein chain contained in the PDB structures initialized by the script `init_db.py`. All the information retrieved is obtained from the mappings between PDB and UniProt generated by the script `make_db.py` through SIFTS.

Attributes		
Name	Type	Description
pdb_id	String	ID of the PDB structure
left_id	String	ID of the left chain in alignment
right_id	String	ID of the right chain in alignment
align_id	String	ID of the alignment
has_fasta	Bool	Whether alignment is available as FASTA file
fasta_height	Int	Number of sequences in the FASTA file
fasta_width	Int	Number of residues in the query sequence in the FASTA file
has_distance	Bool	Whether distance matrix is available
avg_distance	Float	Average value of distance matrix
std_distance	Float	Standard deviation of distance matrix
has_contacts	Bool	Whether contact maps are available
num_contacts	Int	Area of the contact map
all_contacts	Int	Number of heavy atom contacts
cab_contacts	Int	Number of carbon atom contacts
has_predicts	Int	Whether predictions are available
predicts_mi	Int	Whether mutual information matrices are available
predicts_psi	Int	Whether PSICOV estimates are available

Table 2.8: Summary alignments table

Table 2.8 exposes the attributes of the alignments table created by the script `scan_db.py`. Each row contains the identifier of the PDB structure and the identifiers of both the protein chains in the alignment. In case of intra-chain alignments the left and right chain identifiers are equal. Features of the FASTA alignment file are retrieved from the file itself. Instead, information about the PSICOV and mutual information estimated contact maps are retrieved from the respective HDF5 formatted files.

Chapter 3

Results

In this chapter, the results obtained by executing the pipeline described in Section 2.2 are presented. First of all, the features of the initial dataset are described. Since such dataset undergoes modifications as long as it flows through each step in the pipeline, the output of such steps are described as well. This includes an analysis of the alignments in the final dataset, used as input for all the prediction methods.

Secondly, methods based on either mutual information or PSICOV for residue contact predictions are applied on the training partition of the final dataset. In this step, best values for hyper-parameters are found by looking at their behaviour when applied on the training dataset itself. Moreover, such parameters are compared with the theoretical ones, retrieved from the scientific literature closely related to methods taken into account.

Finally, different prediction methods are compared between each other on the test partition of the final dataset. At this stage, such methods are evaluated by looking at the distributions of some fundamental statistics, such as precision, sensitivity and F-score. Such statistics have been chosen in order to limit the influence of true negative values in confusion matrices, since the latter are obtained by using contact maps as ground truth. In addition, the effects of different characteristics of the input alignments on the distribution of chosen statistics are analyzed. Moreover, when evaluation prediction outcomes, we take into consideration that methods based on mutual information have different and less stringent requirements on input alignments, with respect to PSICOV.

3.1 Retrieved linearly interacting protein structures

Initial dataset was generated starting from two datasets, namely FLIPPER and DIBS. Both these datasets contain predictions for LIP residues over protein chains inside various PDB structures. DIBS can be considered as a subset of FLIPPER, since it contains a smaller number of predictions. Predictions stored in DIBS define whether the entire protein chain is linearly interactive or not. Instead, predictions on FLIPPER define whether a specific region in the protein chain is linearly interactive or not. Therefore,

FLIPPER is considered to be more precise with respect to DIBS in the the prediction of linearly interacting residues.

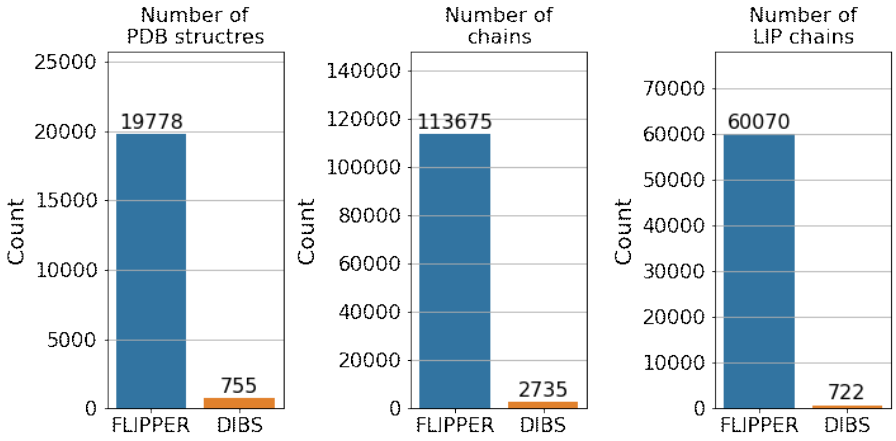


Figure 3.1: Number of items in initial dataset

As Fig. 3.1 shows, initial FLIPPER and DIBS datasets contain LIP predictions over 60070 and 722 different chains, respectively. Since two or more different chains can belong to the same structure in PDB, this leads to a total amount of 19778 different protein structures in the first dataset and 755 in the second one. Considering all such protein structures, the total number of chains raises to 113675 in FLIPPER dataset and to 2735 in DIBS.

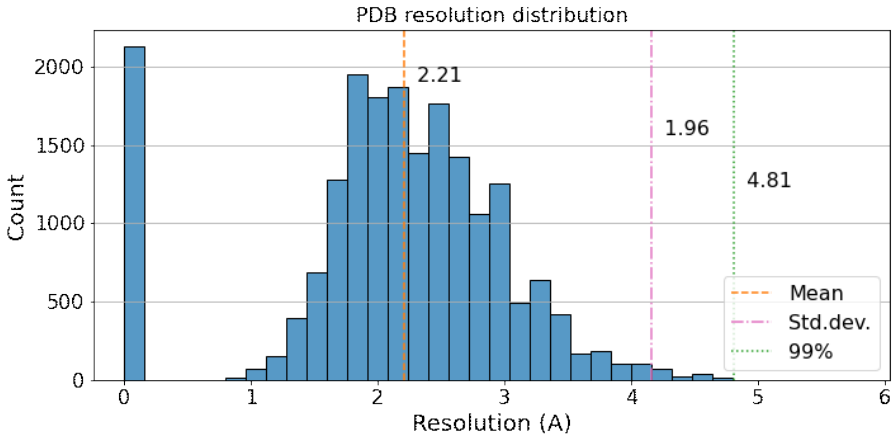


Figure 3.2: Distribution of the resolution of the protein (PDB) structures.

The histogram in Fig. 3.2 represents the distribution of the resolution of all the unique protein structures contained in FLIPPER. Outliers are defined as all those structures having a resolution exceeding the 99th percentile. Among those outliers, there are structures whose resolution is above 50Å. However, the majority of the PDB structures have been observed with a resolution value lower than 5Å.

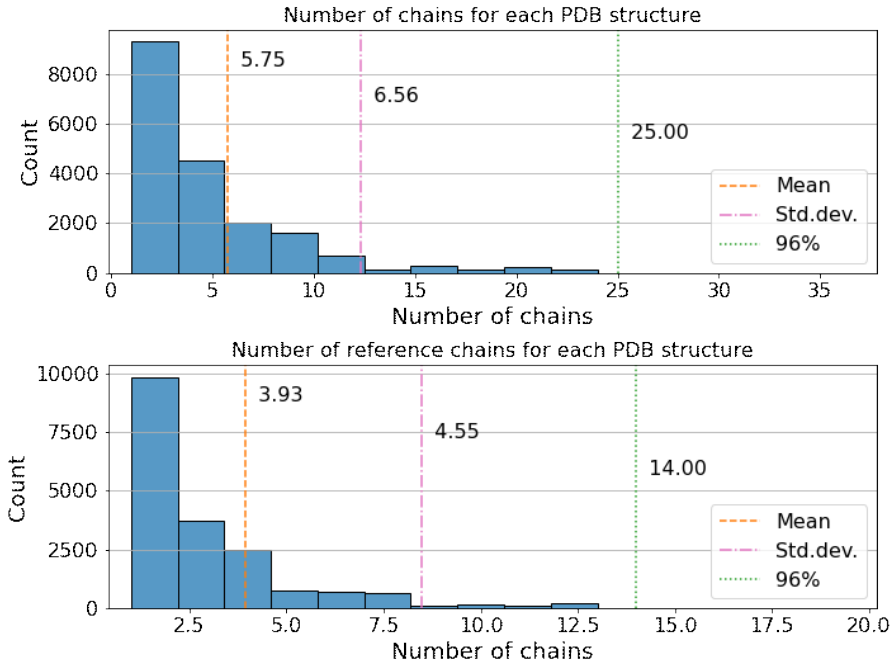


Figure 3.3: Distribution of the number of chains in each PDB structure

Fig. 3.3 shows the distribution of the number of chains belonging to all the protein structures in FLIPPER. The upper histogram considers all the chains, independently form their similarity. The mean number of chains for each PDB structure is around 6. Furthermore, the majority of the PDB structures contain less than 25 chains. Instead, the lower histogram takes into account only one among a group of protein chains having the same amino acidic sequence, here defined as reference chains. As expected, there are fewer reference chains for each PDB structure, with a mean value around 4 and the 99th percentile falling at 14.

All chains in each protein structure are mapped to one or more UniProt entries through SIFTS. However, only the UniProt entry which better covers the amino acidic sequence of the protein chain is considered in this setting, making it a one-to-one relationship. The upper histogram in Fig. 3.4 shows the distribution of the number of chains to which each UniProt entry is mapped. Instead, the histogram on the bottom shows the number of reference chains mapped to each UniProt entry. This means that there are on average 6 chains for which an alignment can be produced out of a UniProt query sequence.

The number of residues which make a protein chains are referenced as its size. Hence, histograms in Fig. 3.5 show the distribution of the chain's size contained in PDB structures within FLIPPER. It must be noticed that distributions have been computed considering only reference chains. This allows to exclude the effect that multiple identical protein chains could have on the distribution itself. Starting from the top, the first distribution takes into account all reference chains. In the second distribution only

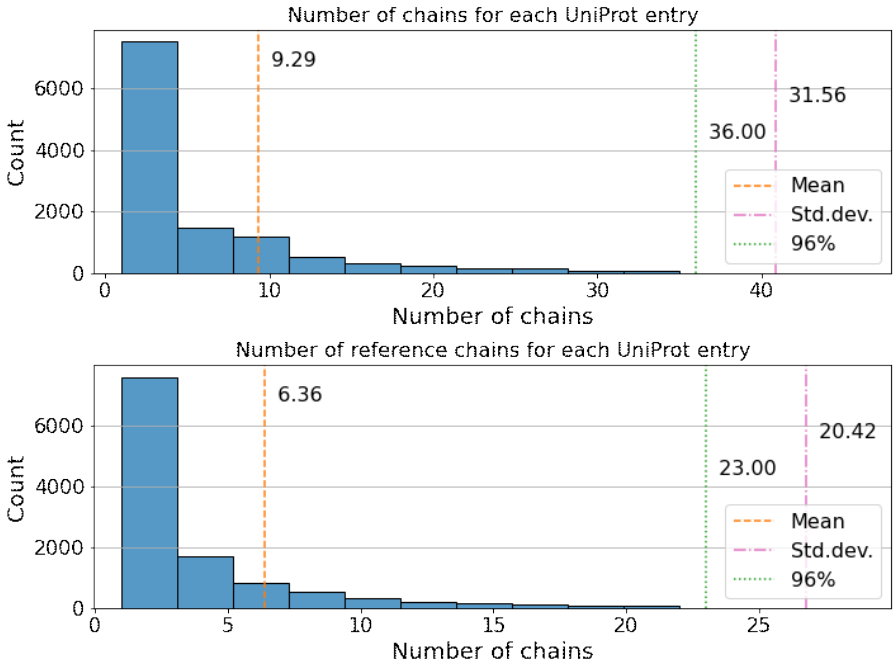


Figure 3.4: Distribution of the number of chains for each UniProt entry

reference chains for which no residue has been predicted as linearly interacting have been considered. Conversely, only reference chains with linearly interacting peptides were taken into account by the last distribution in the figure. As expected, non-LIP chains tend to be larger than LIP chains, showing an higher mean value with respect to the latter.

Developing distance matrices and true contact maps could be computationally expensive. Therefore, to avoid bottlenecks that increased the computation time for this step as it was empirically observed, a subset of all the protein structures contained in FLIPPER was defined. The subset has been determined by looking at the distribution of the statistics showed in the figures above.

Among all the PDB structures in the initial FLIPPER database, a partially random subset of 5000 has been selected. The decision rules used to define such subset are the following:

1. The resolution of each PDB structure must be lower or equal than 5.0Å, according to the 99th percentile in Fig. 3.2 and to the minimum distance threshold for contacts, which is 6.0Å in case of heavy atoms;
2. The number of chains in each PDB structure must be lower or equal than 10. This upper bound is useful to avoid computing more than $10^2 = 100$ distance matrices and 200 contact maps;
3. The number of reference chains in each PDB structure must be between 2 and 5, both included. This upper bounds the number of alignments to be computed for

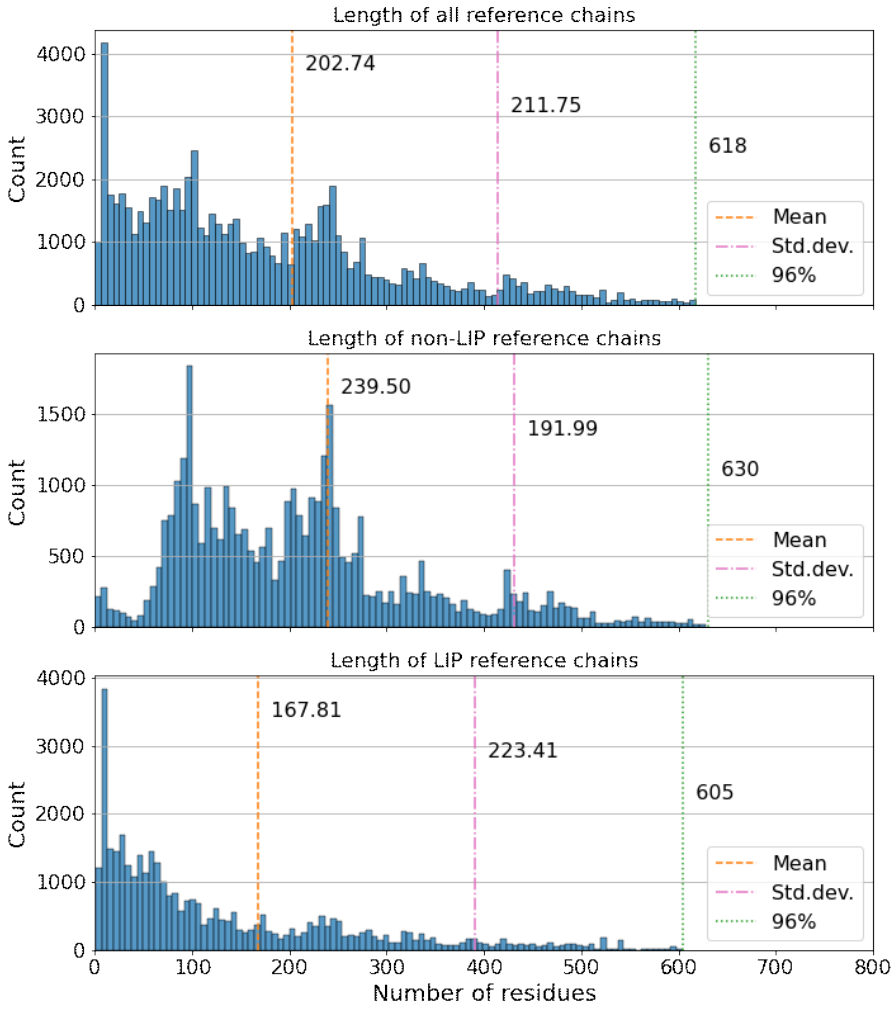


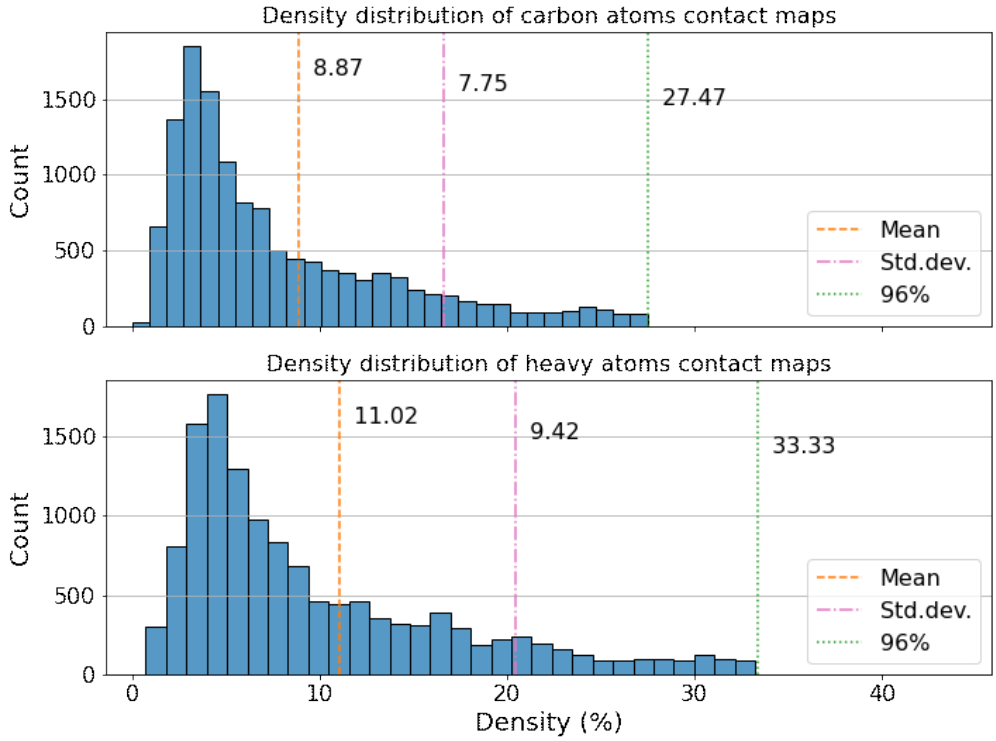
Figure 3.5: Distribution of the width for protein chains, also called chain width.

each PDB structure to $5 \cdot 4 = 20$;

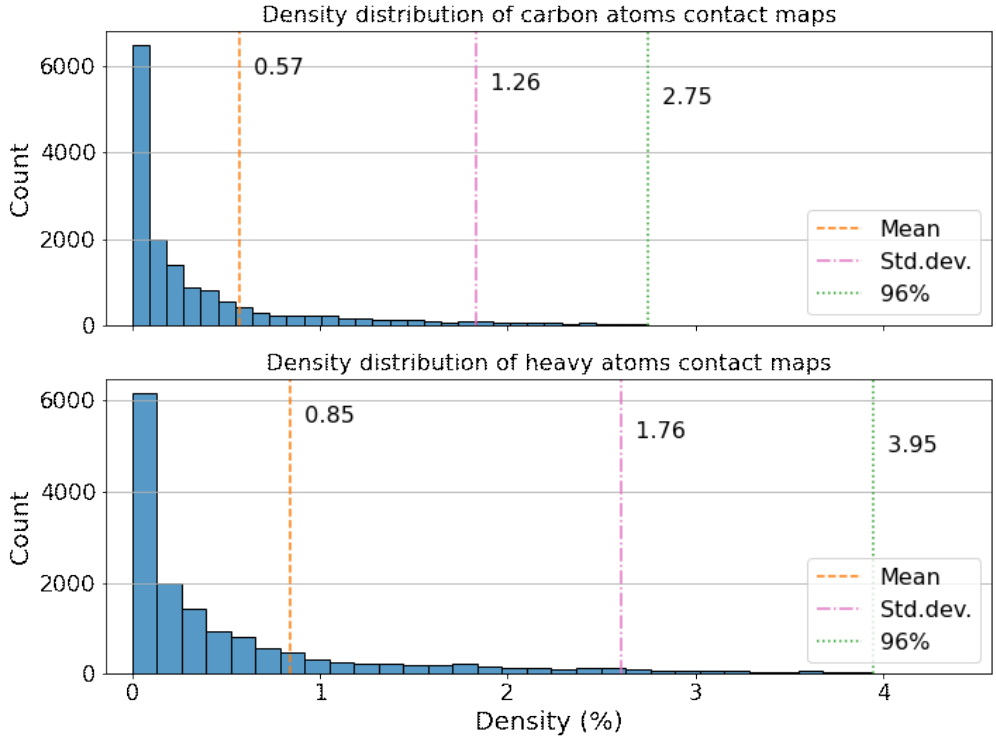
4. The minimum number of residues in a chain of each PDB structure must be greater or equal than 20, while the maximum must be lower or equal than 2000, according to distributions in figure Fig. 3.5. This allows to contain the size of the input alignment on which predictions must be performed, partially avoiding memory leak issues;

5. Structures in DIBS have the priority: the list of PDB structures has been shuffled and then the ones that belong to DIBS have been selected first. In this way, all the structures satisfying all the other properties and belonging to DIBS are selected, while the remaining entries in the subset are taken from FLIPPER.

Fig. 3.6(a) contains two histograms. Both the histograms present the density computed on intra-chain contact maps. In the first one, density has been computed between



(a) Distribution of the density of true contact maps for intra-chain alignments



(b) Distribution of the density of true contact maps for inter-chain alignments

carbon atoms. In the second one, density has been computed between heavy atoms instead.

Fig. 3.6(b) is similar to Fig. 3.6(a) but it contains density computed for inter-chain contact map. By comparing histograms for intra-chain and inter-chain alignments it is clear that density is much lower in the latest case. The first reason for this behaviour is that contacts between two different chains are fewer with respect to contacts between residues in the same chain. Moreover, a chain does not make contact with all the other chains in the same PDB structure, pushing the distribution towards zero.

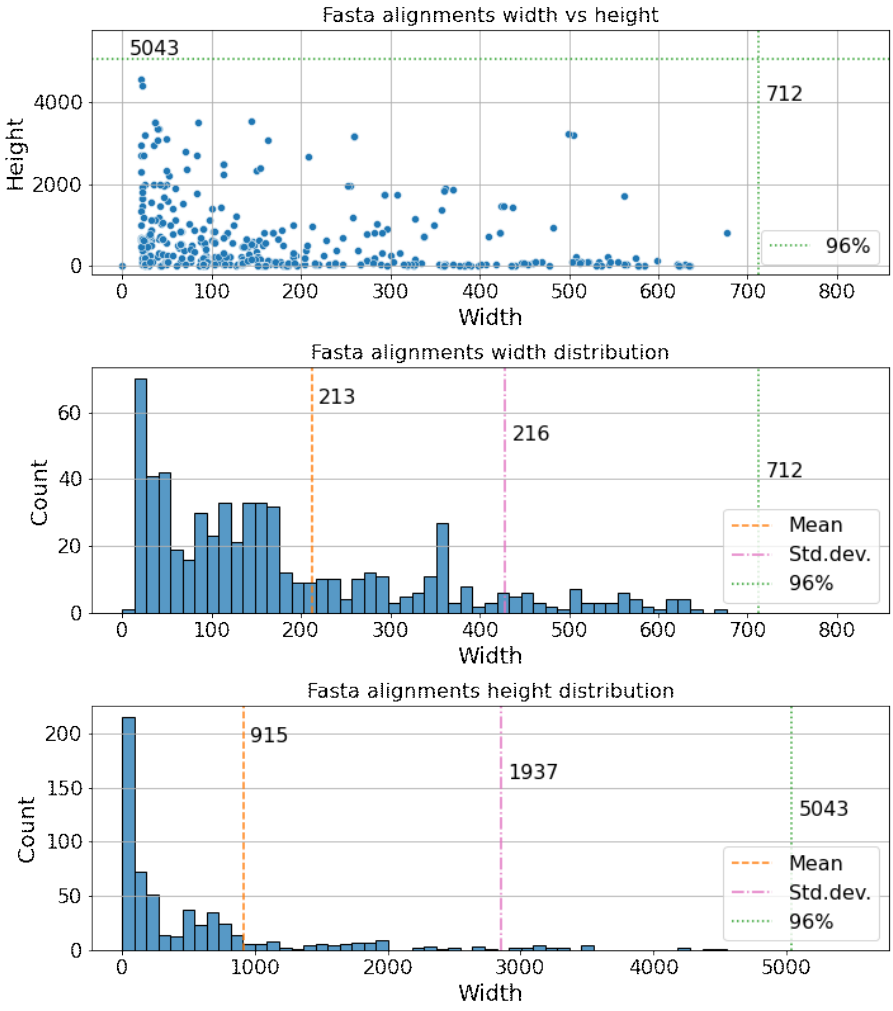


Figure 3.7: Distribution of the size of input alignments.

As already mentioned in the description of the pipeline in Section 2.2.4, generating the multiple sequence alignments is the most resource demanding step, since it requires to scan the UniProt database. Moreover, after each multiple sequence alignment has been generated, it needs to be mapped back to one or two concatenated protein chains in the same PDB structure and finally clustered. These steps are executed within the same script, adding computational burden.

Therefore, to limit the number of multiple sequence alignments which must be generated, a subset of 100 PDB structures was defined starting from the 5000 ones for which distance matrices and contact maps were computed beforehand. Such smaller subset has been defined following the only rule of prioritizing PDB structures contained in both FLIPPER and DIBS datasets, rather than the ones contained only in the former one.

Fig. 3.7 shows information about the FASTA files obtained from the subset of 100 PDB structures defined above. The scatterplot in the first row shows the relationship between the number of residues and the number of sequences in each alignment. The number of residues and the number of sequences in an alignments are also referenced as its width and height, respectively. The last two histograms show the distribution of the number of residues and the number of sequences in the alignments, respectively. The alignments taken into account were already clusterized through the Hobohm algorithm mentioned in Section 2.2.5. Hence, most of the alignments produced have 213 residues and 915 sequence clusters on average. Furthermore, most of them contain fewer than 712 residues and 5043 sequence clusters.

Once that multiple sequence alignments in FASTA format were generated for all the chain pairs in each selected PDB structure, each of them has been assigned to either the training or test dataset through a 60/40 partitioning. Then, all alignments have been fed as input to contact prediction methods based on either mutual information or graphical lasso. However, not all alignments are guaranteed to produce a valid output, especially when PSICOV is applied.

Fig. 3.8 shows the number of alignments associated to either one or two concatenated sequences. In the whole dataset there are 455 different input alignments, of which 241 are intra-chain and 214 are inter-chain. Prediction methods based on mutual information retrieve valid results for all the 241 (100%) intra-chain input alignments but only for 208 (97%) inter-chain ones. Instead, PSICOV retrieves valid results for only 185 (77%) intra-chain input alignments and 41 (19%) inter-chain ones.

It can be observed that PSICOV struggles to converge on inter-chain alignments, given the low number of result it produces in this case. Inter-chain alignments are composed of two concatenated protein chains. This leads to large alignments, which makes convergence too slow to converge within the given time interval.[13]

The distribution of the width of the alignments is presented in Fig. 3.9. In the first plot, distributions take into account all the multiple sequence alignments, divided in intra-chain and inter-chains ones. Instead, the second and the third plots take into account the training and test partitions respectively. Methods based on mutual information were able to retrieve an output for almost all input alignments. This is the reason why the distribution of the alignments' width for those methods resembles the overall one. Instead, it can be noticed that PSICOV reached convergence on alignments with a smaller number of residues in all the partitions of the input dataset.

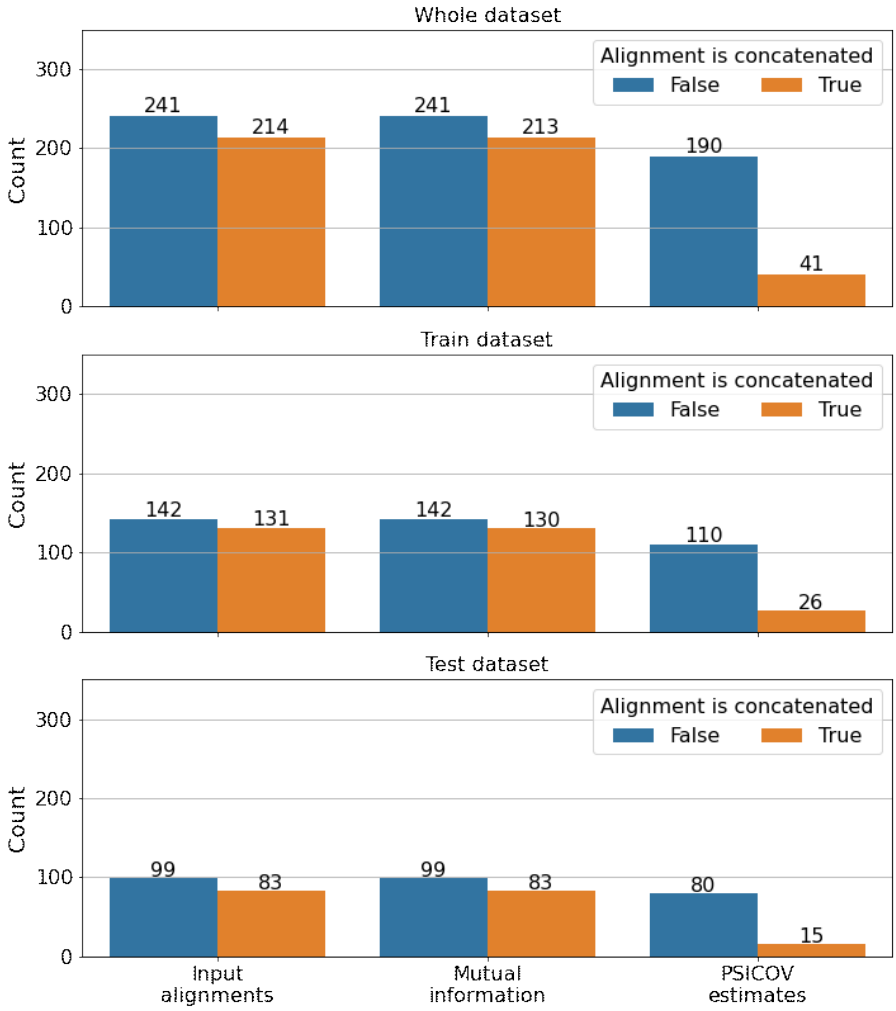


Figure 3.8: Number of output obtained through different contact prediction methods

In Fig. 3.10 the distribution of the height of the alignments can be observed. The first plot show the distribution within the whole dataset. Instead, the second and the third show the distribution for both its partitions. Again, mutual information produced a result for each of the input alignments, the distribution of the alignments' height for this method matches the overall one. Instead, PSICOV was able to reach convergence only on alignments with an higher number of sequence clusters.

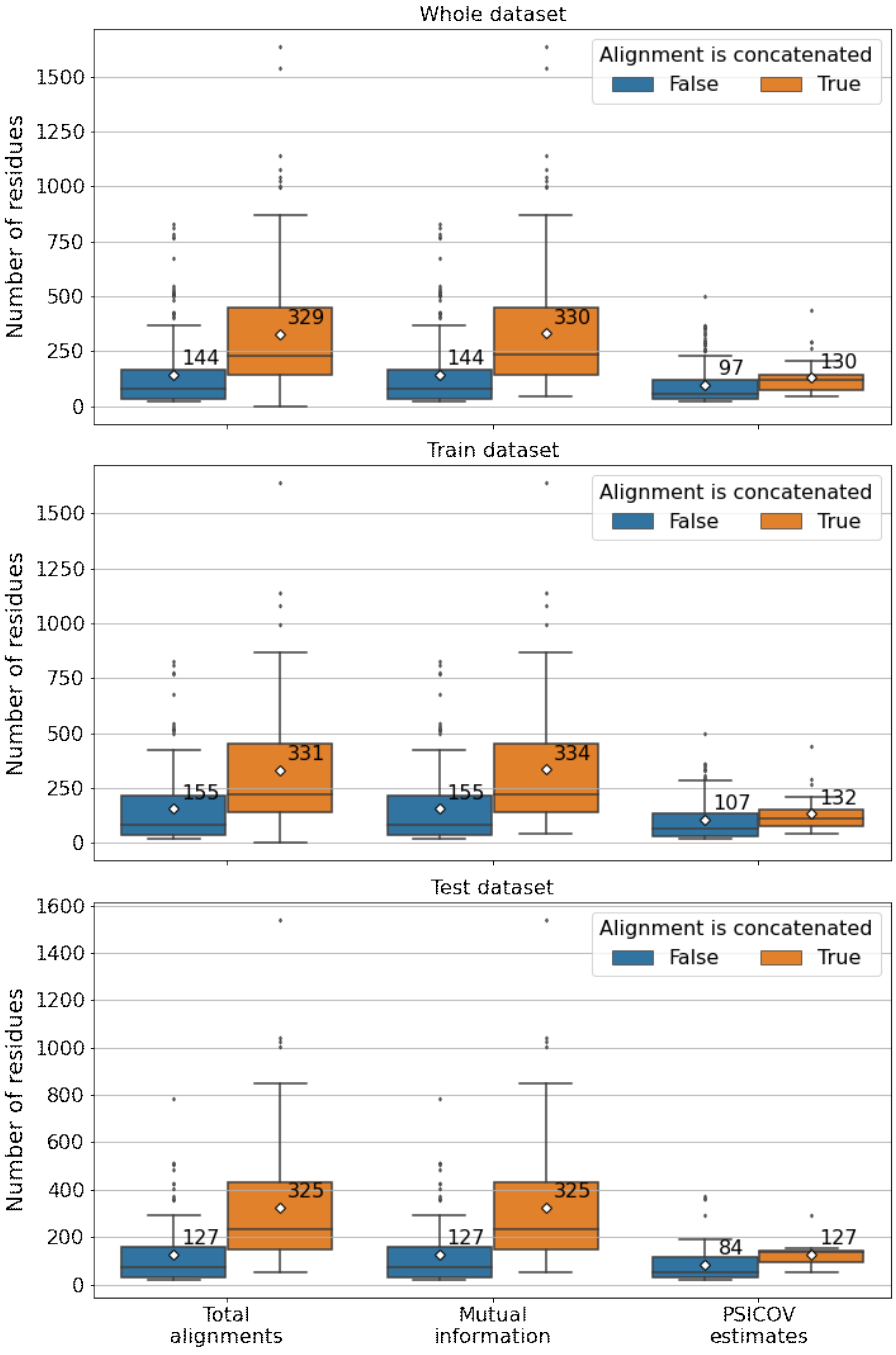


Figure 3.9: Distribution of the width of the alignments

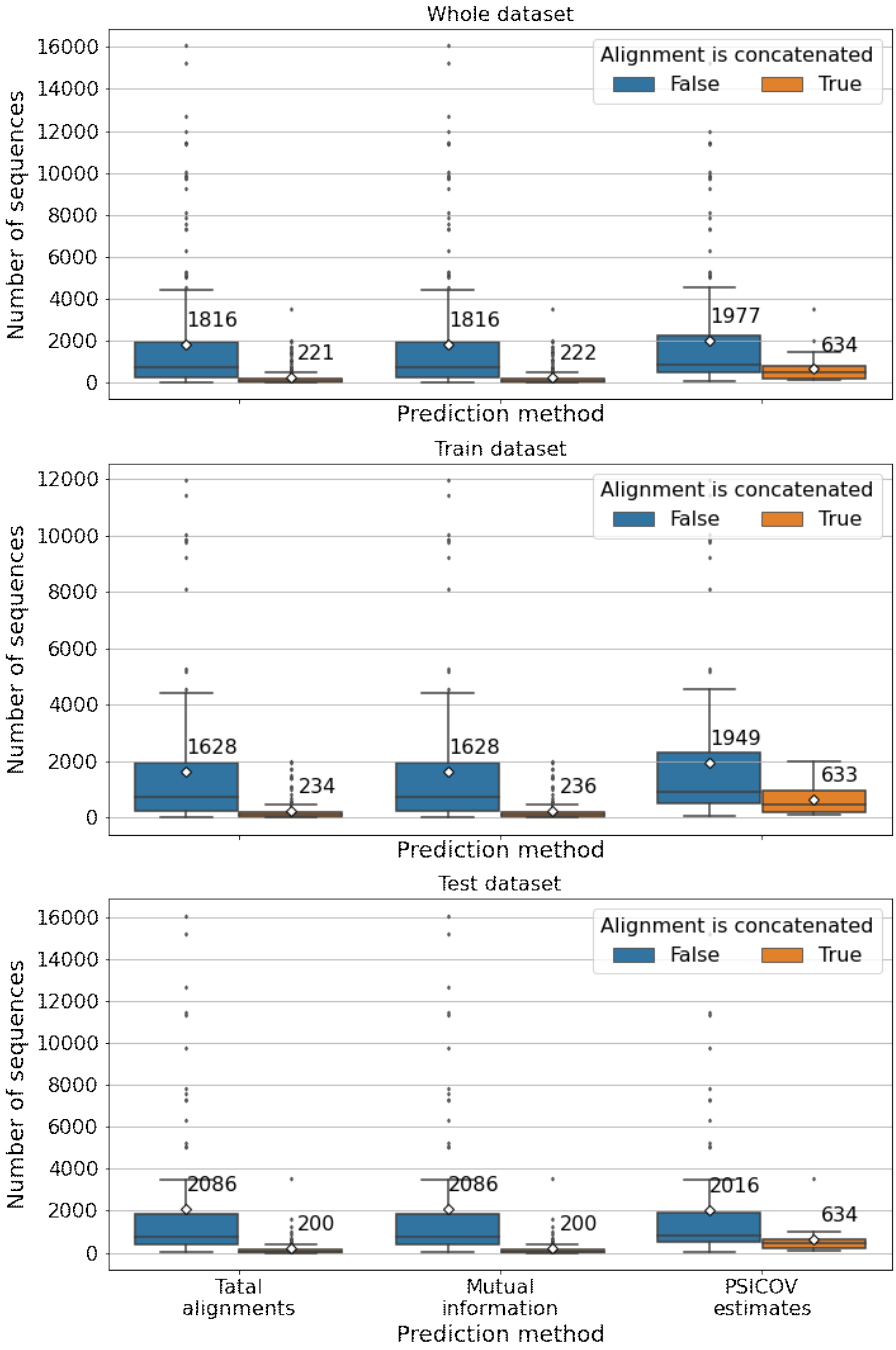


Figure 3.10: Alignments height distribution.

3.2 Choice of the best threshold for prediction methods

In this section all the different contact prediction methods are applied on the training dataset. As explained in the previous Section 3.1, the training dataset contains 268 alignments for which mutual information could be computed. Among these alignments, 142 are intra-chain and 126 are inter-chain. Taking into account PSICOV, the number of alignments for which it produced a valid output in the training dataset drops to 136, of which 110 are intra-chain and 26 are inter-chain.

Both the methods based on mutual information and PSICOV estimate contact maps. Contact maps are matrices whose values could be different according to either the input fed to the prediction methods or the methods themselves. For example, while mutual information takes value inside the interval $[0, 1]$, corrected mutual information assumes values in $\mathbb{R}$.

Furthermore, intra-chain and inter-chain contact maps are different. The former are squared, symmetrical and diagonal. Since contacts between close residues are guaranteed, a band of 6 residues is excluded around the diagonal when evaluating intra-chain contacts prediction. In addition, symmetry allows to select only upper triangular contact maps in such setting. Instead, inter-chain contact maps are not symmetrical nor diagonal and the size of their edges matches the size of the chains involved. They can be retrieved either as the upper right or lower left sub-matrix in a prediction involving two concatenated chains. This means that inter-chain contact maps contain the furthest points from the diagonal, making it a harder target to predict with respect to intra-chain ones.

Therefore, it is important to find a threshold function to define which predicted values should be considered as positive contact predictions and which ones should not, instead. To define such threshold function, the distribution of the values of all predicted contact maps were taken into account. In addition, prediction results are strongly dependent on the input alignment. Hence, threshold function could be influenced by the characteristics of the alignments themselves, e.g. the number of their sequences, the number of residues in their query sequence or whether they were generated from a single sequence or two concatenated ones. These features were included in the analysis for the identification of the best threshold function.

There are various scores that can be computed to compare prediction methods. Scientific literature linked to mutual information and PSICOV often uses precision, also known as Positive Predictive Value (PPV). Given the number of true positives (TP) and false positives (FP) in a confusion matrix, precision can be computed as follows:

$$PPV = \frac{TP}{TP + FP} \tag{3.1}$$

Precision allows to pursuit maximization of true positives without taking into account residues which are not in contact. Contact maps have are guaranteed to have

low density, typically around 3%. Therefore, maximizing the number of true negatives would lead to predicting all the residues pairs as not in contact.

Another way of maximizing the number of true positives without taking into account true negative values is through sensitivity, also known as recall, hit rate or True Positive Rate (TPR). Considering false negatives (FN) in a confusion matrix, sensitivity can be computed as follows:

$$TPR = \frac{TP}{TP + FN} \tag{3.2}$$

Both precision and sensitivity can be considered together by considering the F_β-score. In the F_β-score, the parameter β defines how many times sensitivity is significant with respect to precision. F_β-score is 1.0 when both precision and sensitivity are maximized. Instead, when at least of these two values is close to 0.0, it will tend to 0.0. For both methods, precision has been chosen to be four times as important as sensitivity, i.e:

$$F_\beta\text{-score} = \frac{(1 + \beta^2) \cdot \text{PPV} \cdot \text{TPR}}{\beta^2 \cdot \text{PPV} + \text{TPR}} \quad \text{with} \quad \beta = 0.25 \tag{3.3}$$

3.2.1 Choice of threshold values for mutual Information

The prediction methods based on mutual information which have been applied on the training dataset have been described in Section 1.7.2. They are mutual information *(MI)*, entropy normalized mutual information *(MI/H)* and both APC *(MI-APC)* and ASC corrected mutual information *(MI-ASC)*.

For each method, a contact map has been computed for every alignment in the training dataset. Then, 30 simulations were made with randomization over rows in the alignment as suggested in Dunn, Wahl, and Gloor [10]. Simulations have been used to define a sample mean and standard deviation of results of methods applied on alignments without co-evolving sites. Hence, Z-score has been computed using such statistics. Therefore, Z-score can be used as a test for defining a significant predicted value over a pair of residues. The number of simulations has been chosen empirically in order to limit the time required to compute the time taken by this step.

In Fig. 3.11, the distribution of the Z-scores is shown for each prediction method based on mutual information in the range $[-10, 10]$, i.e. within ten standard deviation in absolute value. Being them histograms, the frequency of a given Z-score is on the vertical axis, while the value of the Z-score itself is on the horizontal axis. Given a prediction method, the blue line represents the distribution of the Z-scores for all the training alignments. Instead, the orange and the green lines represent the distribution of the Z-scores for pairs of residues in contact, defined between carbon and heavy atoms respectively. It can be observed that such distribution for residues in contact is moved to the right with respect to one considering all residues pairs. This behaviour is particularly emphasized in the latest two plots, for both APC and ASC corrected mutual information. Therefore, it should be possible to define a threshold over Z-scores

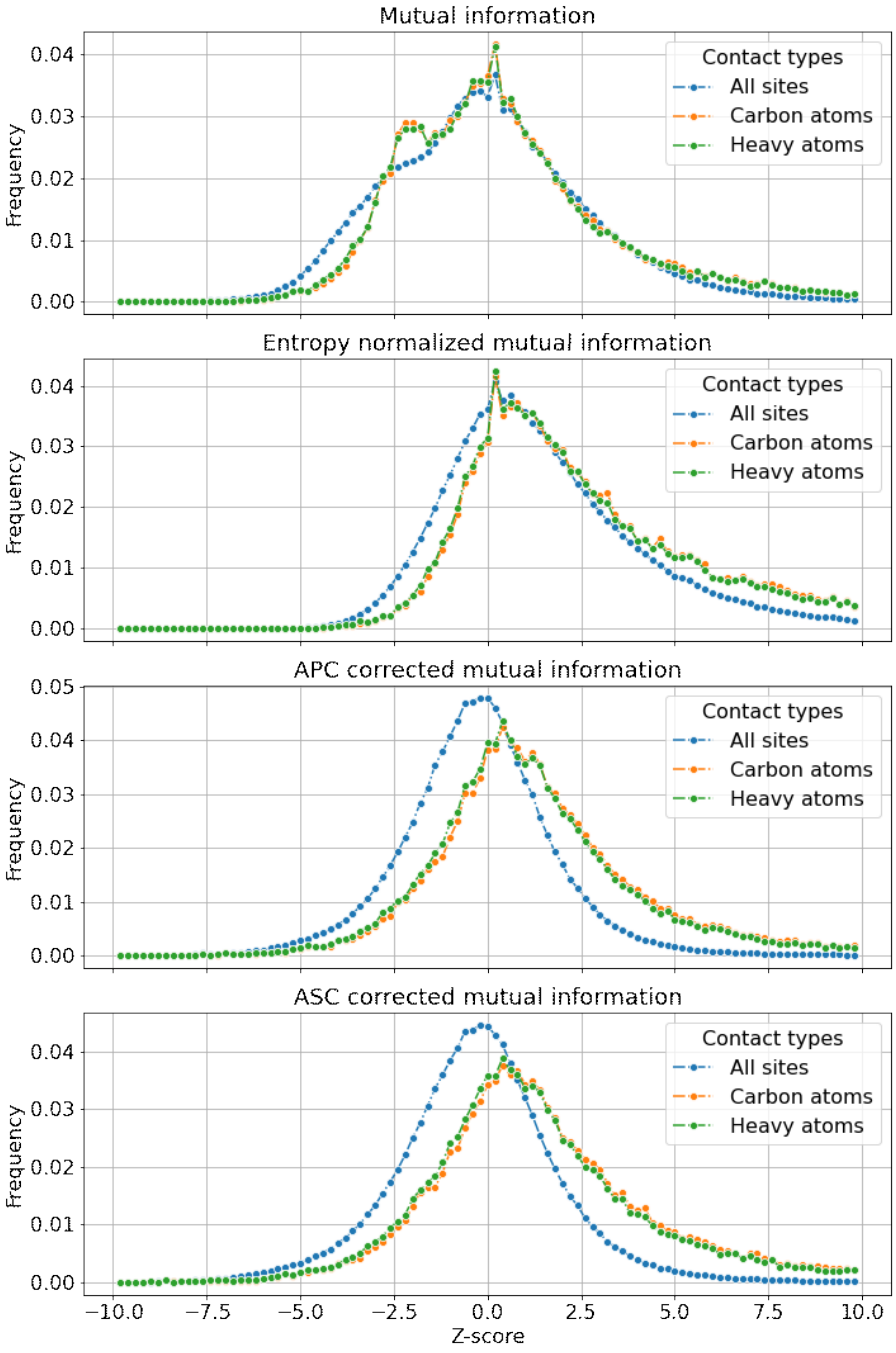


Figure 3.11: Distribution of the Z-scores for different prediction methods applied on the training dataset.

such that the number of true positive contact predictions is maximized through a score such as precision.

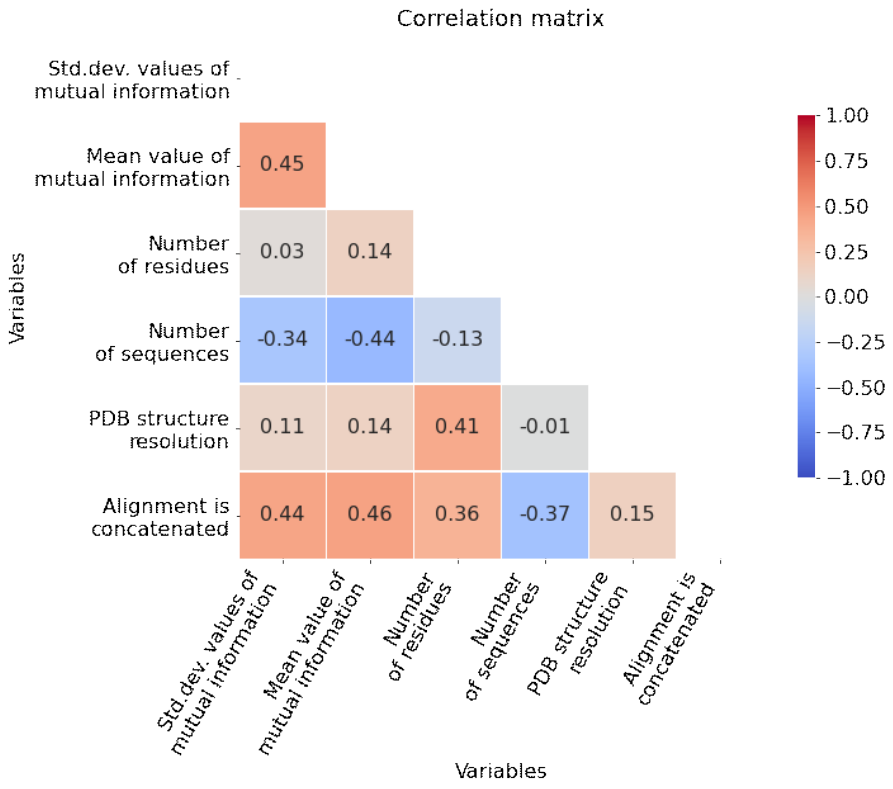


Figure 3.12: Correlation matrix between both the features of the input alignments and the statistics of mutual information.

Finding a suitable threshold for contact prediction methods based on mutual information depends strongly on the distribution of the latter. Thus, it is important to investigate the influence which some features of the input alignments exert on it. The correlation matrix in Fig. 3.12 was computed between the features of the input alignments and some statistics of the mutual information distribution. Such statistics are the mean and standard deviation of non standardized mutual information.

The first feature which seems to have a relevant influence on the mean mutual information value is the height of the alignment. Confusion matrix suggests that the value for mutual information decreases as as the number of sequences in the alignment increases. Fig. 3.13 clearly shows that mutual information decreases rapidly as the number of sequences in the alignment increases.

Another feature which appears to be relevant for the value of mutual information is the fact that an alignment is concatenated or not. Correlation matrix in Fig. 3.12

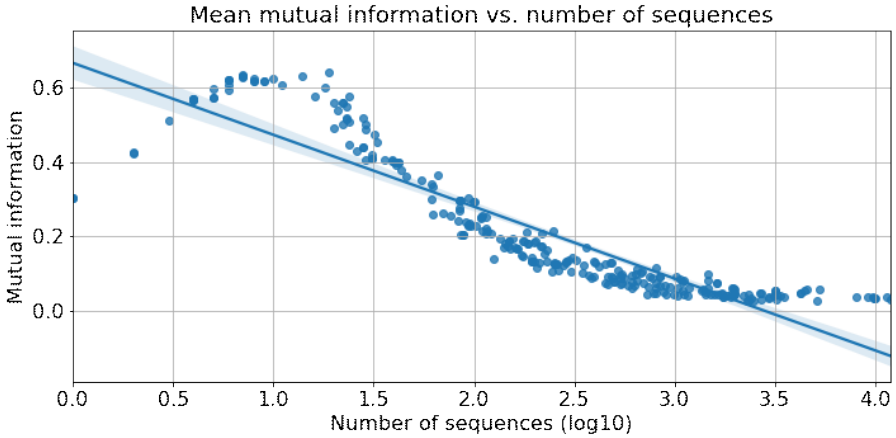


Figure 3.13: Variation of average standardized mutual information with the width of the alignment.

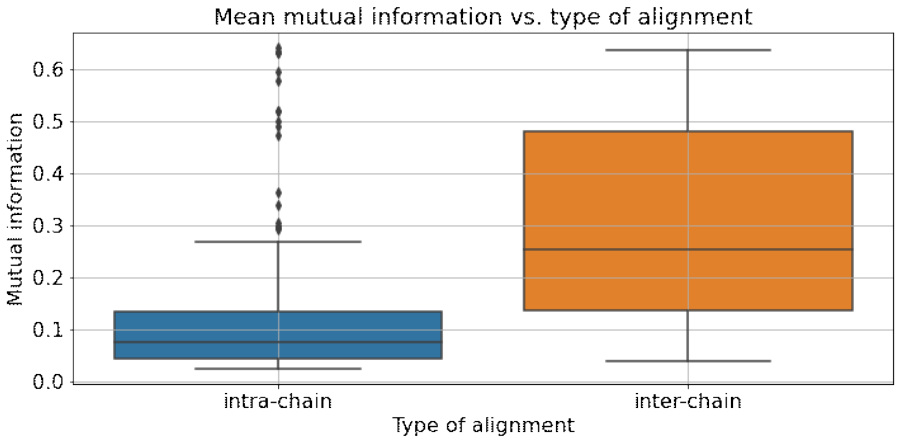


Figure 3.14: Distribution of average mutual information for intra-chain and inter-chain alignments, separately.

shows that mutual information is higher for inter-chain alignments with respect to intra-chain ones. This behaviour is confirmed by the plot in Fig. 3.14. Hence, parameters for intra-chain and inter-chain contact predictions have been defined separately to exploit such difference.

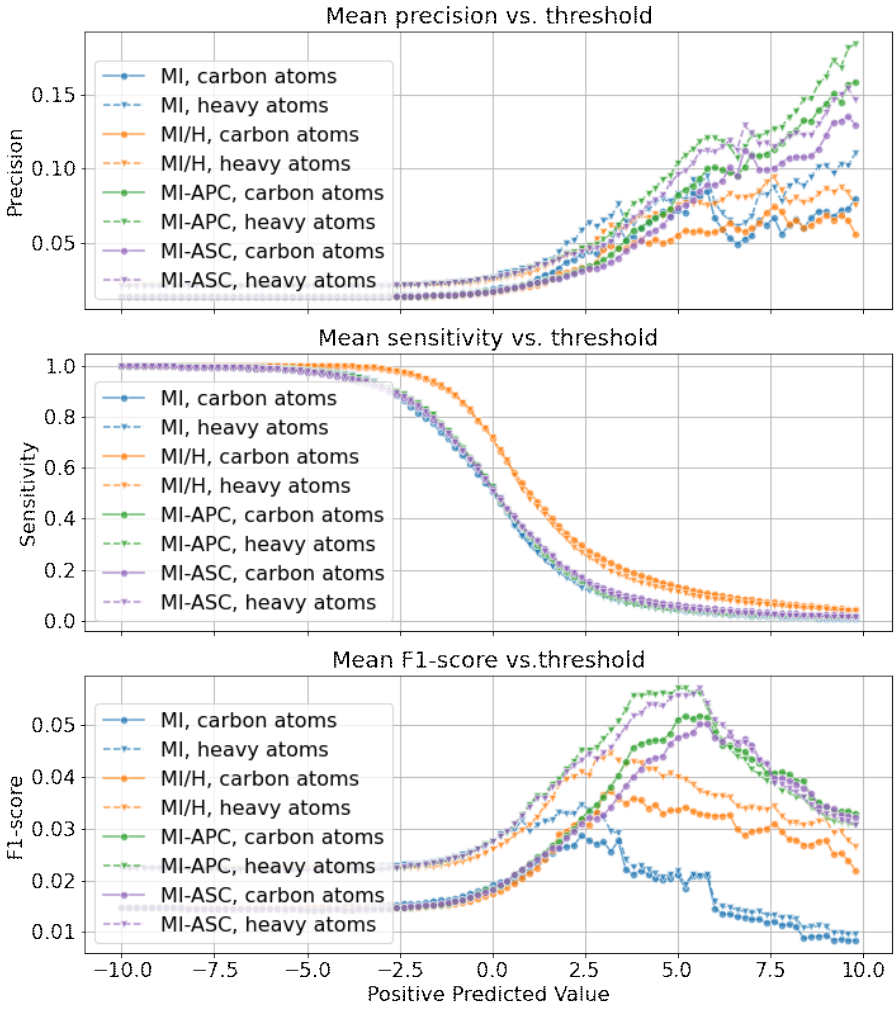


Figure 3.15: Mean precision, sensitivity and F1-score for contact prediction methods based on mutual information applied on the training set.

In Fig. 3.15, the mean value of various statistics is plotted against the chosen threshold values. These statistics are precision, sensitivity and F_β-score and are shown in the first, second and third plots respectively. Mean values have been chosen over single values computed for each threshold value in order to exclude influence of the size of the contact maps. One hundred Z-scores has been evaluated as threshold values taking value in the interval $-10, +10$ standard deviations.

By looking at the plot for $F_\beta - score$, APC and ASC corrected mutual information methods score the best, followed by entropy normalized mutual information. The same

observations can be made about precision. Instead, sensitivity is higher for entropy normalized mutual information.

				Average stat.			Conf. matrix		
Pred. method	Contact map	Is concat.	Thres. value	PPV	TPR	F_β-score	TP	FP	FN
MI	$C_{\alpha\beta}$	False	2.4	7%	13%	4%	7448	573248	27330
MI	$C_{\alpha\beta}$	True	5.8	6%	3%	2%	162	47160	5774
MI	H	False	2.4	9%	12%	5%	10433	570263	39420
MI	H	True	2.6	2%	14%	2%	1461	409703	8085
MI/H	$C_{\alpha\beta}$	False	3.2	8%	24%	6%	14384	812046	20394
MI/H	$C_{\alpha\beta}$	True	3.6	1%	19%	1%	1033	527799	4903
MI/H	H	False	3.2	11%	20%	7%	19888	806542	29965
MI/H	H	True	3.6	2%	16%	2%	1670	527162	7876
MI-APC	$C_{\alpha\beta}$	False	5.6	17%	6%	9%	3876	41697	30902
MI-APC	$C_{\alpha\beta}$	True	5.8	2%	3%	1%	86	17860	5850
MI-APC	H	False	5.2	20%	5%	10%	5630	48339	44223
MI-APC	H	True	3.8	2%	4%	2%	353	77324	9193
MI-ASC	$C_{\alpha\beta}$	False	5.8	17%	7%	8%	4469	52622	30309
MI-ASC	$C_{\alpha\beta}$	True	5.6	2%	3%	1%	116	27891	5820
MI-ASC	H	False	5.6	20%	6%	10%	6137	55668	43716
MI-ASC	H	True	3.8	2%	5%	2%	397	92545	9149

Table 3.1: Best parameters for methods based on mutual information, computed on training partition

The choice of the best parameters has been made for intra-chain and inter-chain mutual information matrices separately. Table 3.1 shows the chosen threshold values which maximizes mean F_β-score, for each method, type of contact and type of alignment.

3.2.2 Choice of threshold values for PSICOV

PSICOV produces covariance estimates by means of graphical lasso. It has various hyper-parameters, already presented in Section 1.7.6. Most of those parameters were fixed beforehand for the algorithm to be sufficiently permissive, hence retrieve the largest number of contact predictions within the given time limit.

Specifically, ρ parameter defining the convergence threshold was set to 0.001 as suggested by the author, low-scoring filters were activated, minimum occupancy threshold was set to 0.8, and sequence separation was set to 6.0 to avoid predicting contacts on the backbone. Moreover, positive predicted values were required instead of raw scores, i.e. values in the interval $[0, 1]$ defining the likelihood of a pair of sites to be in contact.

Even if using the parameters described above were quite permissive, the number of alignments for which PSICOV successfully estimated the covariance matrix is 136 on the total of 273, as shown in Fig. 3.8. In the same figure, it can be also observed that the number of estimations is unbalanced towards the intra-chain alignments, with respect to the inter-chain ones. This, and the distribution of the width of the alignments for which PSICOV predictions were successfully generated in Fig. 3.9, highlights the slower convergence of PSICOV when applied to wider alignments.

The density of the target contact map to be predicted by PSICOV is the only parameter which has been fine-tuned.. Given two different protein chains A, B within the same structure, the target contact map will be computed comparing residues in the concatenated chain $C = A + B$ between themselves. Instead, considering a single chain A the contact map is computed by comparing only its residues among each other.

Since setting correctly the density of the target contact maps could influence positively the estimation of covariance, correlation matrix in Fig. 3.16 was developed. In such correlation matrix, various features of the input alignments were compared with the target density of the contact maps computed either between carbon atoms or heavy atoms. In both cases, low correlations were found, with a maximum score of 0.11 between the density of heavy atoms contact maps and the width of the alignment.

In Fig. 3.17, an alignment being intra-chain or inter-chain was used as a proxy for alignment width. However, no significant variation in the distribution can be observed. Mean density value is 2% for contact maps computed between carbon atoms, while it is 3% for the ones computed between heavy atoms. 3% was chosen as target density for PSICOV estimates.

Fig. 3.18 shows the distribution of the positive predicted values through PSICOV. It is an histogram: the predicted values themselves are displayed on the horizontal axis, while the relative frequencies are displayed on the vertical axis. The line in blue represents the distribution of all the predicted values greater than 0.0. Instead, green and orange lines represent the same distribution taking into account only contacting residue pairs, define between either carbon or heavy atoms.

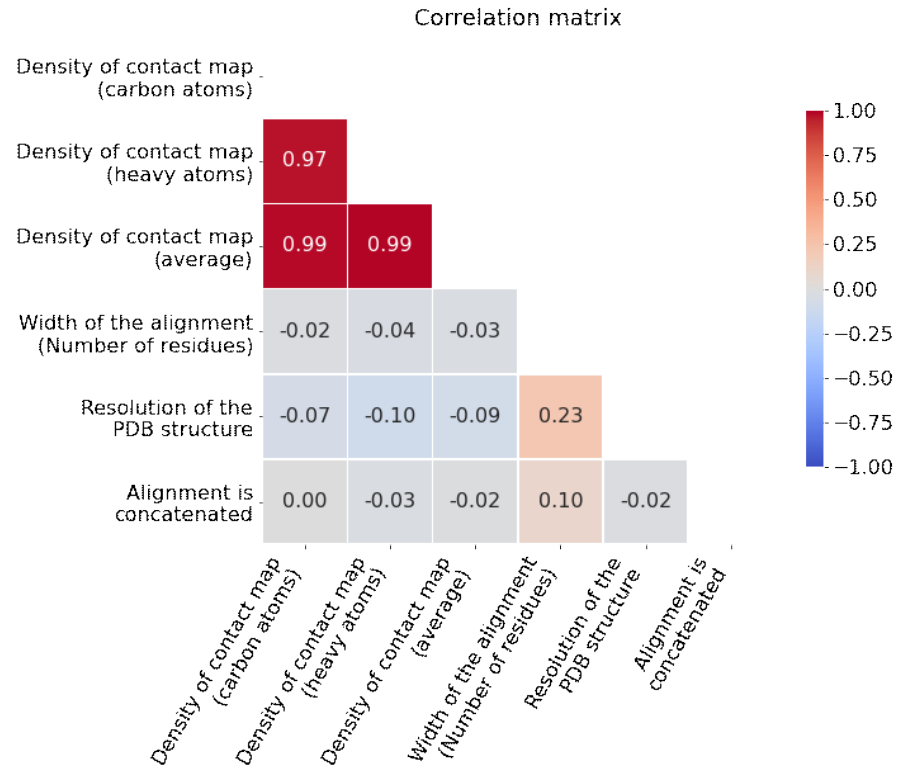


Figure 3.16: Correlation matrix between the features of the input alignments

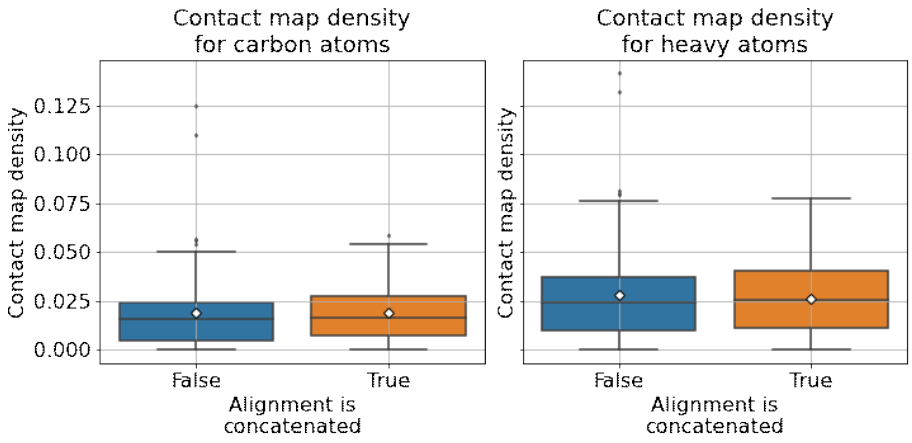


Figure 3.17: Distribution of the target contact map density computed between carbon atoms or heavy atoms, conditioned by the the type of input alignment

Taking into account the distribution in figure Fig. 3.18, a threshold value had to be defined over positive predicted values in order to obtain boolean predictions. Given

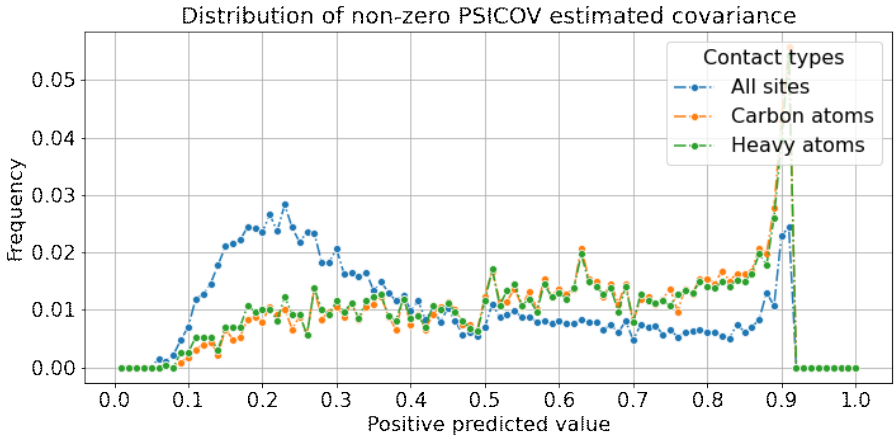


Figure 3.18: Distribution of the positive predicted values.

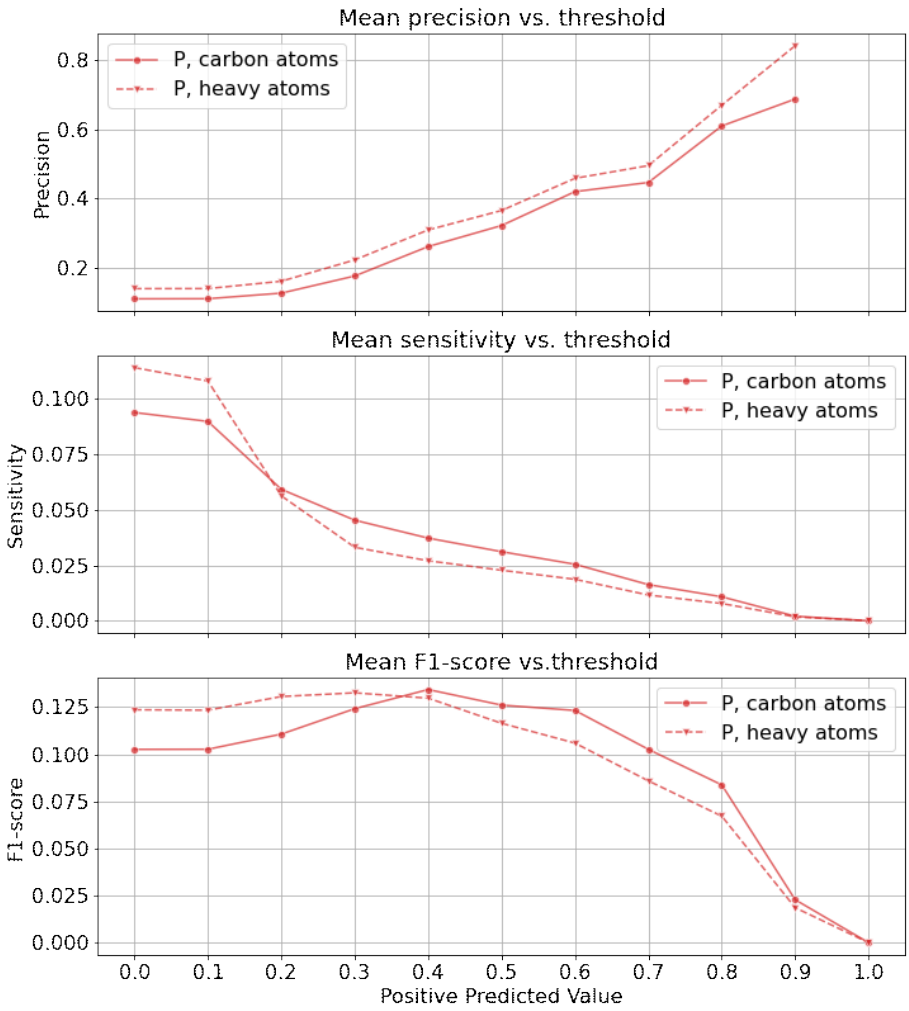


Figure 3.19: Distribution of the positive predicted values.

that the positive predicted values range in the interval $[0.0, 1.0]$, ten evenly spaced threshold values have been considered in Fig. 3.19.

As expected, mean precision increases as the threshold value increases. Considering either contacts computed between carbon or heavy atoms, mean precision exceeds 50% after a predicted positive value of 0.7. Instead, the value for sensitivity is always very low. It is around 1% when all residues pairs with a positive predicted values over 0.0 are considered to be in contact. Then, it rapidly decreases toward 0% as the value for the threshold increases. Lastly, the value of the $F_\beta$score reaches its highest peak between 0.3 and 0.4 predicted values.

				Average stat.			Conf. matrix		
Pred. method	Contact map	Is concat.	Thres. value	PPV	TPR	F_1-score	TP	FP	FN
P	$C_{\alpha\beta}$	False	0.6	52%	2%	15%	1187	706	16496
P	$C_{\alpha\beta}$	True	0.3	6%	3%	4%	21	466	938
P	H	False	0.6	57%	2%	13%	1342	551	24605
P	H	True	0.3	6%	3%	4%	26	461	1429

Table 3.2: Best parameters for PSICOV.

In Table 3.2 the best threshold values are listed. The table contains also the mean statistics obtained for all the alignments in the training dataset. Moreover, the sum of all the confusion matrices generated by looking at prediction results are made available. By looking at the number of both true and false positives in concatenated alignments, it can be noticed that they are almost one thousandth with respect to the same values for intra-chain alignments. This underlines the difficulty in applying this method on inter-chain alignments, since the contacts which must be predicted are far away from the backbone where predictions tend to accumulate.

3.3 Evaluation of contact prediction methods

Both methods based on mutual information and PSICOV have been evaluated against each other on the test partition. First, each prediction method has been considered separately. Statistics have been computed on all the contact predictions retrieved through it. In this setting, PSICOV produces better results with respect to the ones based on mutual information when looking to precision. However, it must be noticed that the number of results obtained through PSICOV are fewer, especially when only inter-chain alignments are considered as input.

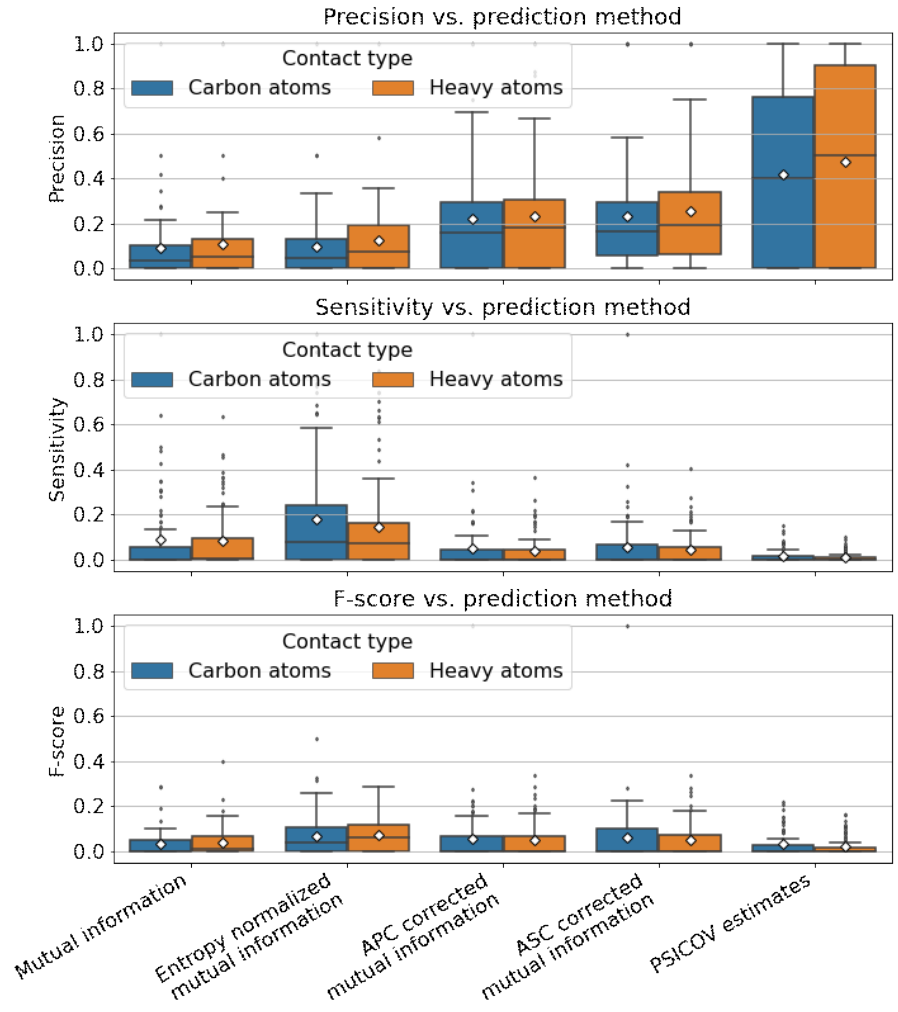


Figure 3.20: Distribution of precision, sensitivity and F_β-score on the test set

Fig. 3.20 shows precision, sensitivity and F_β-score for every method involved in the analysis. In this case, only alignments for which both PSICOV and method mutual information produced a result have been taken into account. Again, methods based on mutual information perform worse than PSICOV. Among these, APC and ASC corrected mutual information have the highest average precision.

Furthermore, methods based on mutual information have a higher mean value for precision on this subset rather than on the whole test dataset. In the first case mean precision is around 10% for both APC and ASC corrected mutual information, while in the latter it exceeds 20%. This suggests that PSICOV produced results for alignments with higher quality, on average.

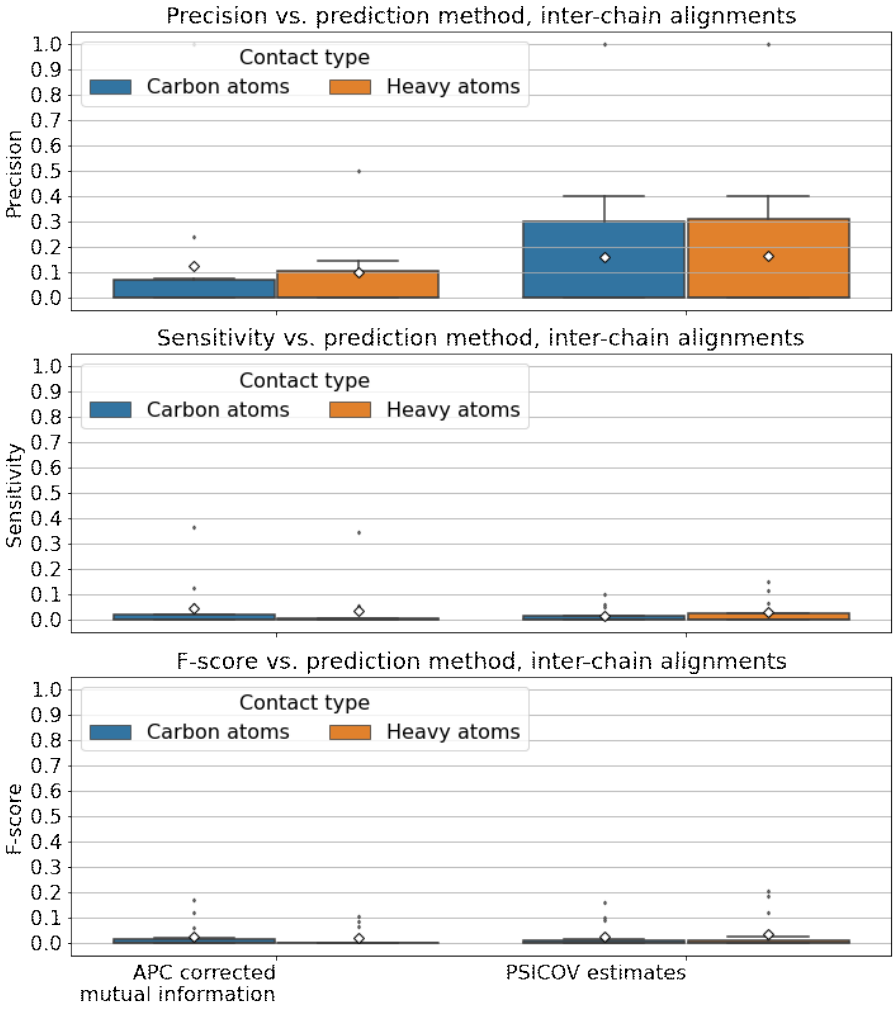


Figure 3.21: Distribution of precision, sensitivity and F_β-score on inter-chain alignments of the test set

The distribution of the statistics for contact predictions sightly changes when intra-chain and inter-chain alignments are considered separately. Statistics showed in Fig. 3.21 have been calculated only on alignments for which both PSICOV and APC corrected mutual information were able to predict contact maps. In this case, the precision of the former method is around 15%, while it drops around 10% for the latter. Sensitivity is sightly below 10% for both methods and so does F_β-score.

Precision observed in Fig. 3.20 for APC corrected mutual information shows a mean value resembling the one presented for the best 100 predictions in Jones et al.

[13] and Dunn, Wahl, and Gloor [10]. In the first case, intra-chain alignments were generated through three iterations of Jackhmmer [11] against the UniProt dataset, while in the latter high quality alignments were borrowed from Pfam. Instead, the behaviour of the statistics observed in Fig. 3.21 shows clearly that predicting contacts between two different chains is harder than predicting contacts within residues of the same chain. Moreover, it suggests that producing high quality input alignments play a key role in contact predictions. However, this task has proven to be difficult, given the statistics observed above.

Chapter 4

Conclusions

A complete pipeline for the prediction and evaluation of both intra-chain and inter-chain contacts prediction has been developed. The pipeline takes as input predictions of linear interacting peptides on protein chains in PDB. It maps information from PDB to UniProt and back by means of SIFTS in order to obtain multiple sequence alignments. Then, it computes true contact maps computed over both carbon atoms and heavy atoms. Finally, it produces contact predictions using methods based on mutual information or PSICOV and evaluates them against the associated contact maps.

Usually, benchmark datasets containing good quality alignments are used to evaluate methods for intra-chain chain contacts prediction. Nevertheless, fewer applications of contacts prediction methods to inter-chain alignments can be found in scientific literature. Even less material has been produced with respect to linear interacting peptides. In this work both tasks of intra-chain and inter-chain contacts prediction have been exploited over sequences containing linear interacting peptides.

In order to limit the computational resources required by the whole pipeline to run in such high dimensional settings defined by multiple sequence alignments, a careful selection of a batch of 100 PDB structures has been carried out. The analysis which led to the definition of such subset took into account various of their features, e.g. their resolution, the number and the size of their chains.

Finally, the results obtained by all the contact prediction methods have been analyzed. The results obtained from intra-chain contacts prediction were similar to the ones found in literature. Instead, the difficulty of inter-chain contact prediction methods has been assessed, given that worse results have been obtained through the same methods.

Among the contact prediction methods involved, PSICOV proved to perform sightly better in both intra-chain and inter-chain settings. However, PSICOV requires more computational resources to converge with respect to the ones required by the computation of the mutual information and its regularizations. This led PSICOV to converge only on alignments with higher quality. However, considering only those alignment it still perform better than any method based on mutual information.

The first reason why the same contacts prediction methods perform better on intra-chain alignments with respect to inter-chain can be found in the model linking primary structures to tertiary and quaternary ones. In fact, co-evolution is much stronger when two residues are in contact within the same protein chain, since those contacts are fundamental for its folding. Instead, residues involved only in inter-chain contacts are subject to fewer functional constraints.

Another reason behind the observed performance of contact predictors lie in the generation of joint alignments. Joint alignments concatenate sequences which are supposed to have an interaction. Such interaction is approximated by the fact that two sequences belong to the same organism. Moreover, only one sequence for each organism has been kept, in order to avoid producing enormous alignments which would be difficult to handle. Both these solutions introduce loss of information. To avoid this issue some other ways for generating joint alignments could be taken, such as the one proposed in the mapping between UniProt and the PDB.

Furthermore, tools such as mutual information and PSICOV could be used as input for a more sophisticated contacts predictor. For example, a meta-estimator involving a deep neural network could be trained by utilizing the whole initial dataset instead of just one batch as done during this work. However, it would require a lot more computational resources.

Bibliography

[1] G. R. et al. “Stereochemistry of polypeptide chain configurations”. In: *J. Mol. Biol.* 7.1 (1963), pp. 95–99. DOI: 10.1016/s0022-2836(63)80023-6.

[2] H. B. et al. “The Protein Data Bank”. In: *Nucleic Acids Res.* 28.1 (2000), pp. 235–242. DOI: 10.1093/nar/28.1.235.

[3] J. D. et al. “SIFTS: updated Structure Integration with Function, Taxonomy and Sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins”. In: *Nucleic Acids Res.* 47.D1 (2019), pp. 235–242. DOI: 10.1093/nar/gky1114.

[4] S. V. et al. “SIFTS: Structure Integration with Function, Taxonomy and Sequences resource”. In: *Nucleic Acids Res.* 41.D1 (2013), pp. D483–D489. DOI: 10.1093/nar/gks1258.

[5] S. A. et al. “Basic local alignment search tool”. In: *J. Mol. Biol.* 215.3 (1990), pp. 403–410. DOI: 10.1016/S0022-2836(05)80360-2.

[6] S. A. et al. “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs”. In: *Nucleic Acids Res.* 25.17 (1997), pp. 3389–3402. DOI: 10.1093/nar/25.17.3389.

[7] C. M. Buslje, J. Santos, J. M. Delfino, and M. Nielsen. “Correction for phylogeny, small number of observations and data redundancy improves the identification of coevolving amino acid pairs using mutual information”. In: *Bioinformatics* 25.9 (Mar. 2009), pp. 1125–1131. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp135. eprint: https://academic.oup.com/bioinformatics/article-pdf/25/9/1125/532215/btp135.pdf. URL: https://doi.org/10.1093/bioinformatics/btp135.

[8] T. U. Consortium. “UniProt: a worldwide hub of protein knowledge”. In: *Nucleic Acids Res.* 47.D1 (2018), pp. D506–D515. DOI: 10.1093/nar/gky1049.

[9] T. M. Cover. *Elements of Information Theory*. Vol. 1. Wiley, 1991.

[10] S. Dunn, L. Wahl, and G. Gloor. “Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction”. In: *Bioinformatics* 24.3 (Dec. 2007), pp. 333–340. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm604. eprint: https://academic.oup.com/bioinformatics/article-pdf/24/3/333/16883955/btm604.pdf. URL: https://doi.org/10.1093/bioinformatics/btm604.

[11] S. R. Eddy. “Accelerated Profile HMM Searches”. In: *PLOS Computational Biology* 7.10 (Oct. 2011), pp. 1–16. DOI: 10.1371/journal.pcbi.1002195. URL: https://doi.org/10.1371/journal.pcbi.1002195.

[12] U. Hobohm, M. Scharf, R. Schneider, and C. Sander. "Selection of representative protein data sets". In: *Protein Science* 1.3 (1992), pp. 409–417. DOI: https://doi.org/10.1002/pro.5560010313. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/pro.5560010313. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/pro.5560010313.

[13] D. T. Jones, D. W. A. Buchan, D. Cozzetto, and M. Pontil. "PSICOV: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments". In: *Bioinformatics* 28.2 (Nov. 2011), pp. 184–190. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btr638. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/2/184/16908913/btr638.pdf. URL: https://doi.org/10.1093/bioinformatics/btr638.

[14] E. V. Koonin. "Orthologs, paralogs, and evolutionary genomics". In: *Annu. Rev. Genet.* 39.1 (2005), pp. 309–338. DOI: 10.1146/annurev.genet.39.073003.114725.

[15] R. van der Lee, M. Buljan, B. Lang, R. J. Weatheritt, G. W. Daughdrill, A. K. Dunker, M. Fuxreiter, J. Gough, J. Gsponer, D. T. Jones, P. M. Kim, R. W. Kriwacki, C. J. Oldfield, R. V. Pappu, P. Tompa, V. N. Uversky, P. E. Wright, and M. M. Babu. "Classification of Intrinsically Disordered Regions and Proteins". In: *Chemical Reviews* 114.13 (2014). PMID: 24773235, pp. 6589–6631. DOI: 10.1021/cr400525m. eprint: https://doi.org/10.1021/cr400525m. URL: https://doi.org/10.1021/cr400525m.

[16] L. C. Martin, G. B. Gloor, S. D. Dunn, and L. M. Wahl. "Using information theory to search for co-evolving residues in proteins". In: *Bioinformatics* 21.22 (Sept. 2005), pp. 4116–4124. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti671. eprint: https://academic.oup.com/bioinformatics/article-pdf/21/22/4116/485803/bti671.pdf. URL: https://doi.org/10.1093/bioinformatics/bti671.

[17] A. M. Monzon, P. Bonato, M. Necci, S. C. Tosatto, and D. Piovesan. "FLIPPER: Predicting and Characterizing Linear Interacting Peptides in the Protein Data Bank". In: *Journal of Molecular Biology* 433.9 (2021), p. 166900. ISSN: 0022-2836. DOI: https://doi.org/10.1016/j.jmb.2021.166900. URL: https://www.sciencedirect.com/science/article/pii/S0022283621000942.

[18] *PDB-101*. https://pdb101.rcsb.org//.

[19] W. R. Pearson. "An Introduction to Sequence Similarity ("Homology") Searching". In: *Curr. Protoc. Bioinformatics* 42.1 (2013), pp. 311–318. DOI: 10.1002/0471250953.bi0301s42.

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

[21] E. Schad, E. Fichó, R. Pancsa, I. Simon, Z. Dosztányi, and B. Mészáros. "DIBS: a repository of disordered binding sites mediating interactions with ordered proteins". In: *Bioinformatics* 34.3 (Oct. 2017), pp. 535–537. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btx640. eprint: https://academic.oup.com/bioinformatics/article-pdf/34/3/535/25117200/btx640.pdf. URL: https://doi.org/10.1093/bioinformatics/btx640.

[22] J. R. H. Tame and B. Vallone. "The structures of deoxy human haemoglobin and the mutant Hb Tyrα42His at 120K". In: *Acta Crystallographica Section D* 56.7 (July 2000), pp. 805–811. DOI: 10.1107/S0907444900006387. URL: https://doi.org/10.1107/S0907444900006387.

[23] *UniProt*. https://www.uniprot.org//.

[24] K. R. Wollenberg and W. R. Atchley. "Separation of phylogenetic and functional associations in biological sequences by using the parametric bootstrap". In: *Proceedings of the National Academy of Sciences* 97.7 (2000), pp. 3288–3291. ISSN: 0027-8424. DOI: 10.1073/pnas.97.7.3288. eprint: https://www.pnas.org/content/97/7/3288.full.pdf. URL: https://www.pnas.org/content/97/7/3288.