# Abstract

***Aim:*** This project aims to improve protein domain identification using context information, more specifically domain adjacency and ordering.

***Background:*** Domains are the evolutionary components of proteins. Most proteins are formed by multiple domains and these domains appear in very specific combinations. Domains can be used to infer the functional role of a protein and the idea that there are a limited number of domain combinations that work properly together led to the idea that domain co-occurrence can be used to improve the performance of the existing domain detection methods. In this thesis, we go one step beyond the co-occurrence concept and give evidence to support the hypothesis that domain adjacency and ordering makes more statistical and biological sense than just domain co-occurrence.

***Method:*** We formulate six scoring frameworks based on the adjacency principle which fulfil our requirements of being statistically founded, computationally fast and straightforward to integrate in the existing frameworks.

***Conclusion:*** The results demonstrate that incorporating context, in the form of domain adjacency is statistically significant and can improve domain prediction. The best performing scoring functions had two main features: (a) scaling the adjacency counts with the number of domains in each protein, (b) giving more weight to whichever adjacent domain had a more significant E-value. We also implemented an algorithm that is fast (and therefore tractable on 60 million sequences).

***Contribution:*** This area of research is at its early stage and there are only two other papers that use context to predict protein domains. Nevertheless, our thesis brings several improvements to the existing literature. First, it is the only scoring framework using adjacency and ordering of domains rather than co-occurrence. This is well justified by the the protein domain architecture statistics. Second, it is the first time that superfamily level annotations are used rather than family level annotations. Also, using context to identify superfamilies makes more sense as superfamilies are less conserved than families. Third, our project draws its conclusions based on testing on much more representative data than any of the similar papers. Our method is tested on more than 2000 representative organisms or 60 million protein sequences while existing papers use maximum 8 organisms. There are also other technical points that need to be highlighted. First, our scoring framework has an analytical formulation that makes it statistically sound and fast to compute. Second, we developed the framework such that it can be easily integrated in the existing SUPERFAMILY analysis pipeline.

# Acknowledgments

I would like to thank my supervisors Dr Julian Gough for his continuous support with this research and for introducing me to the field of Bioinformatics.

I would also like to express my thanks to Dr Tim Kovacs for accepting my transfer to the MSc in Machine Learning.

I thank Corina for her moral support and for coming with me to the University of Bristol.

# Table of Contents

# 1 Introduction

## 1.1 Background

The last 20 years have seen the rise of bioinformatics as a method of applying computer science to biology. Ever since the Human Genome Project that managed to completely identify and map the human genome in 2003, it has become apparent the need to manipulate massive amounts of data. The conventional approach of biologists and geneticists proved insufficient boosting the creation of the bioinformatics field which pursues the deciphering and management of the biological data. The Next-Generation Sequencing has become the primary tool for genomic analysis as scientists are running a race against time in identifying what is the role of the 31,000 genes found in the human body. However, in order to grasp the concept of genes, a firm understanding of the role each protein has in the life of each organism (be it bacteria or human) is crucial. In this sense, bioinformatics encompasses a set of tools to aid the scientific field to answer biological questions by using computational power.

A protein is defined as a macromolecule formed from chains of amino acids which fold within a 3D structure that defines the biological function of that given protein. Also, proteins are constructed from domains which are evolutionary units of protein structure and function. At the moment, the computational assignment of a protein's structure and function are arguably the most important applications of bioinformatics. A first step in inferring a protein's function would be to perform domain identification which helps improve protein annotations mainly because protein domains are associated with specific functions. Hence, an improvement to protein domain identification can help improve the annotation of new proteins.

This project will focus on domain adjacency or context information and will devise a new scoring scheme and E-value in order to better identify protein domains which will later be used for unseen/not annotated proteins and determine their function within the SUPERFAMILY pipeline.

## 1.2 Challenges of domain identification

Hidden Markov Models are strong tools for identifying protein domains within new protein sequences. The mapping of genomes has increased exponencially in the last 10 years and the current methods seem to miss numerous domains within highly divergent proteins. An improvement of remote protein homology would be a first step in solving the issue. Thinking at a smaller scale, it is simpler to first adjust the sensitivity of the HMM by exploring the possibilities of domains to appear together; hence the reasoning for using domain adjacency for improving the sensitivity of the scores outputted by HMMs. Better domain identification will thus improve both homology detection and new protein annotation.

This project is mainly based on research and will be heavily reliant on the understanding on how the SUPERFAMILY pipeline works. Most of the work, apart from devising the new scoring and E-value functions, will be conducted on the SUPERFAMILY databases and framework. This implies a thorough understanding of the functioning of the HMM models, especially the SAM framework, the current scoring frameworks, domain architecture, domain assignment.

In a nutshell we are trying to improve the HMM method of protein domain identification by incorporating information on domain adjacency. This approach is yet to be tested as the scarce available research concentrates on domain co-occurrence. The main difference between these two approaches is that domain adjacency takes into account the position and orientation of a domain in reference to another; in domain co-occurrence, the domain combinations can work either way. Biologically speaking this is not true; we can have thousands of combinations of domain architecture A-B and only a couple of domain architecture B-A. This is the main problem that we are trying to mitigate and solve by adjusting the scoring and E-value functions. The challenge here is to devise a mathematical function that is plausible from a biological point of view that will keep use context information to improve domain annotation.

## 1.3 **Significance and contribution of the project**

This project aims to improve protein domain identification using context information, more specifically domain adjacency and ordering.

We plan to achieve our aim through the following objectives:

- Generate statistics of protein domain architectures from SUPERFAMILY;
- Formulate several scoring functions based on those statistics;
- Build a benchmark to estimate if there is any improvement over SUPERFAMILY;

The type of this project is investigative with a fair amount of implementation. The main deliverables are:

- An algorithm which enhances prediction of domains using context-dependent priors, optimised and validated on real data.
- Building a valid benchmark based on the data available in the SUPERFAMILY databases.
- Fast and efficient code which should be capable of analysing 60 million sequences.

The sequencing of the human genome ranks as one of the greatest technological achievements of the last century. Since then an enormous amount of data has been generated by the succeeding sequencing efforts. Analysing this data provides new challenges for computer scientists and has the potential of providing new insights for biology and concrete tools for translational medicine.

Protein sequence analysis lies at the core of understanding the biological processes of organisms. Enhancing the predictions of a state of the art analysis framework like SUPERFAMILY, which affects many researchers, even by a small amount, can have a magnified impact on science.

## 1.4 **Structure and organization of the thesis**

Chapter 2 is an extensive part of the thesis that encompasses 4 major parts: first a general introduction to proteins and domains, together with the databases that harbour information on them; the second part links protein homology with domain identification and describes the methodology for improving protein homology and domain annotation; the third part is crucial to the whole thesis and presents a mathematical framework for the scoring backbone which we present in a subsequent chapter; the last part is a general review of the current literature that improves protein domain identification by using context information. In chapter 3 we present the scoring method developed, together with the data used and the 'man made' benchmark that we had to produce. In chapter 4 we present in details the results we obtained, whilst in chapter 5 we conduct an in-depth evaluation of the current work. We dedicate chapter 6 to delineating the avenues for further research. Chapter 7 will conclude this thesis.

# 2 Background

This chapter is aimed at introducing basic knowledge concerning proteins and domains. The chapter will have four main parts. The first part will act as an introduction to bioinformatics and we will briefly discuss the importance of proteins for life maintenance together with structure and functions of these proteins; we will define what domains are and how they construct the protein's architecture; we will also present the Protein Data Bank (PDB) and Structural Classification of Proteins Database (SCOP) based on which the SUPERFAMILY pipeline is constructed on. In the second part we will further introduce the concepts of homologous and remote homologous proteins which represent the reason why we are trying to improve the protein domain prediction; here we will discuss the traditional and new wave methodologies for detecting homologies. The third part will present the basic mathematical framework that we utilized in developing our method. The last part will be a literature review of the current research done on protein domain co-occurrence.

## 2.1 **Proteins and domains**

### 2.1.1 Structure and function

Proteins are biochemical compounds consisting of one or more polypeptides (which are polymer chains of amino acids) which fold in a typical manner in order to perform a biological function. There are 20 different amino acids (residues) in nature, all with different physicochemical properties among which are hydrophobicity, hydrophilicity, charge etc. These amino acids can be rearranged in order to create different proteins with specific biological functions. The folding of the protein structure is based on the amino acid chains.

The protein's structure is explained via the following hierarchical structures:

- Primary structure – represented by the actual linear amino acid sequence;
- Secondary structure – is defined by the hydrogen bonds of the polymer which describe the atomic position in three-dimensional space. In layman's terms, the secondary structure represents the areas of the protein where coiling and folding takes place;
- Tertiary structure – the final three dimensional structure of the protein molecule defined by atomic coordinates;
- Quaternary structure – the structure formed by several protein molecules into a single larger protein or multi subunit complex.

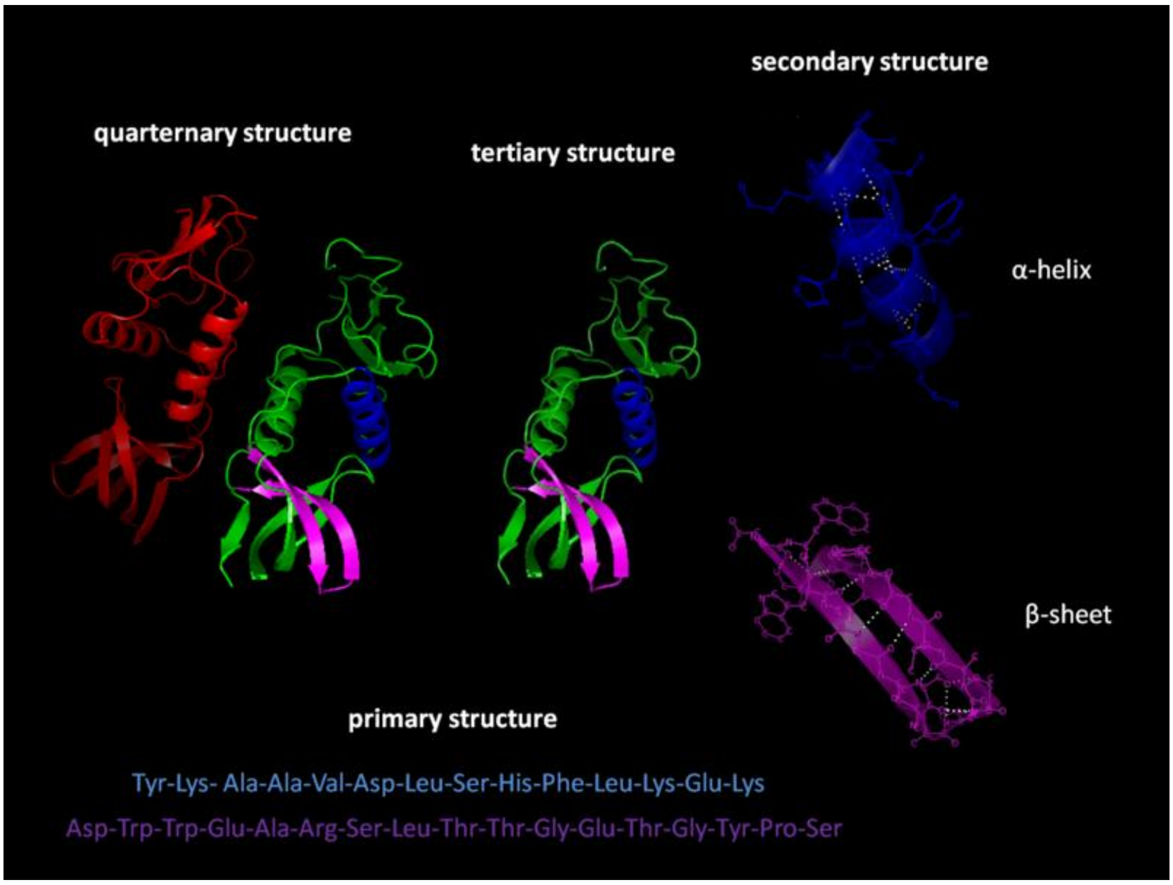

secondary structure
quarternary structure
tertiary structure
α-helix
β-sheet
primary structure
Tyr-Lys- Ala-Ala-Val-Asp-Leu-Ser-His-Phe-Leu-Lys-Glu-Lys
Asp-Trp-Trp-Glu-Ala-Arg-Ser-Leu-Thr-Thr-Gly-Glu-Thr-Gly-Tyr-Pro-Ser

Figure 2-1 Protein structure described by the four hierarchical levels. Souce: http://en.wikipedia.org/wiki/File:Protein_structure.png

Protein molecules perform all the vital cell functions which make proteins the building blocks of life. The functions usually performed by proteins are those of gene regulation, transport of materials, RNA transcription, and act as catalysts for chemical reactions (proteins as enzymes), etc.

### 2.1.2 Domains and protein architecture

Domains are evolutionary units that comprise a protein's structure; the majority of proteins contain more than one domain ([1]). Protein domains can also evolve and function independently from the rest of the protein. They are also responsible for the three dimensional structure of the protein. Since domains are evolutionary units they are the perfect tool for identifying conserved elements within proteins based on sequence homology. Thus, protein domains can be utilized for function association. They are identified in a sequence based on the conservation sites observed in sequence alignments. Domains are a strong utensil for functional identification as conserved elements retain function through evolution.

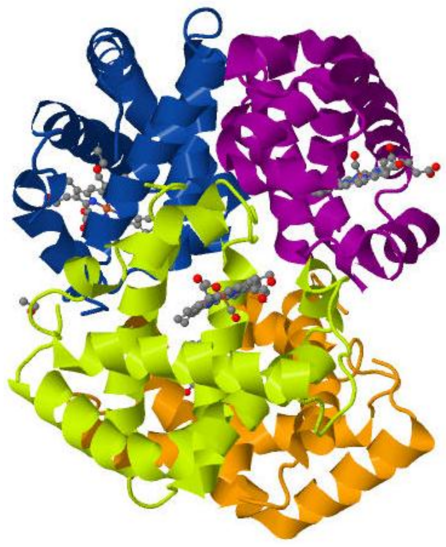


Figure 2-2: 3D rendering of a protein. Its constituent domains are rendered with different colours. Source: Jmol, http://www.rcsb.org/pdb/explore/jmol.do?structureId=1JEB&bionumber=1

Domains can be classified according to the data that defines them. Thus, we have sequence, structure and interacting domains. Sequence domains are derived from the sequence alignments of proteins; structural domains refer to the known 3D structure of a protein domain which can be extracted from a public protein domain database; interacting domains, just as the name suggests are known domains that interact with each other for which the 3D structure is known [2].

Homologous proteins usually share one or more of the same domain. Moreover, domains are regarded as the primary components of proteins and can be recombined or rearranged to create new proteins with different functions. However, not all protein domain combinations are possible in nature; in fact only a small number of two-domain and three-domain combinations are recurrent in proteins ([1]). The authors of [1] define a supra-domain as a domain combination in a certain N-to-C terminal that occurs in at least two different domain architectures in different proteins with either different domains at the N and C terminal end of the combination or different domains at one end and no domains at the other. Moreover, protein domain architecture is defined as a linear combination of protein domains. New proteins can be formed via duplication and recombination in the evolutionary process.

### 2.1.3 Protein Data Bank (PDB)

The Protein Data Bank was established in 1972 as a public repository of protein structures. It is updated weekly and as of September 2012 it contains 69,232 protein structures (http://www.rcsb.org/pdb/home/home.do).

### 2.1.4 Structural Classification of Proteins (SCOP)

SCOP represents a database that classifies all the protein structures within the PDB database. It provides a comprehensive description of the structural and evolutionary relationships of the proteins of known structure [3]. SCOP is manually curated and contains a classification of protein structural domains based on evolutionary relationships; thus, the classification unit is the structural domain. This classification is hierarchical and encompasses 7 levels [3] :

1. Class – arbitrary grouping that describes the types of fold; this is for the user's convenience.
2. Fold – the first level describes the grouping of the fold, whilst this level describes the different shapes of domains within a class.
3. ***Superfamily** – this is the most important level of classification and describes that members of the same superfamily share a common evolutionary ancestor. At this level, the members may not have any more sequence similarity, but preserve a same or similar function.*
4. Family – at this level, members have sequence similarity and often have the same function. At this level, members are grouped on a more recent ancestor due to the sequence similarity.
5. Protein domain – at this level the protein domains are grouped based on the idea they are the same protein.
6. Species – at this level, domains are grouped based on the species they are part of.
7. Domain – at this level we identify those proteins that are uni-domained.

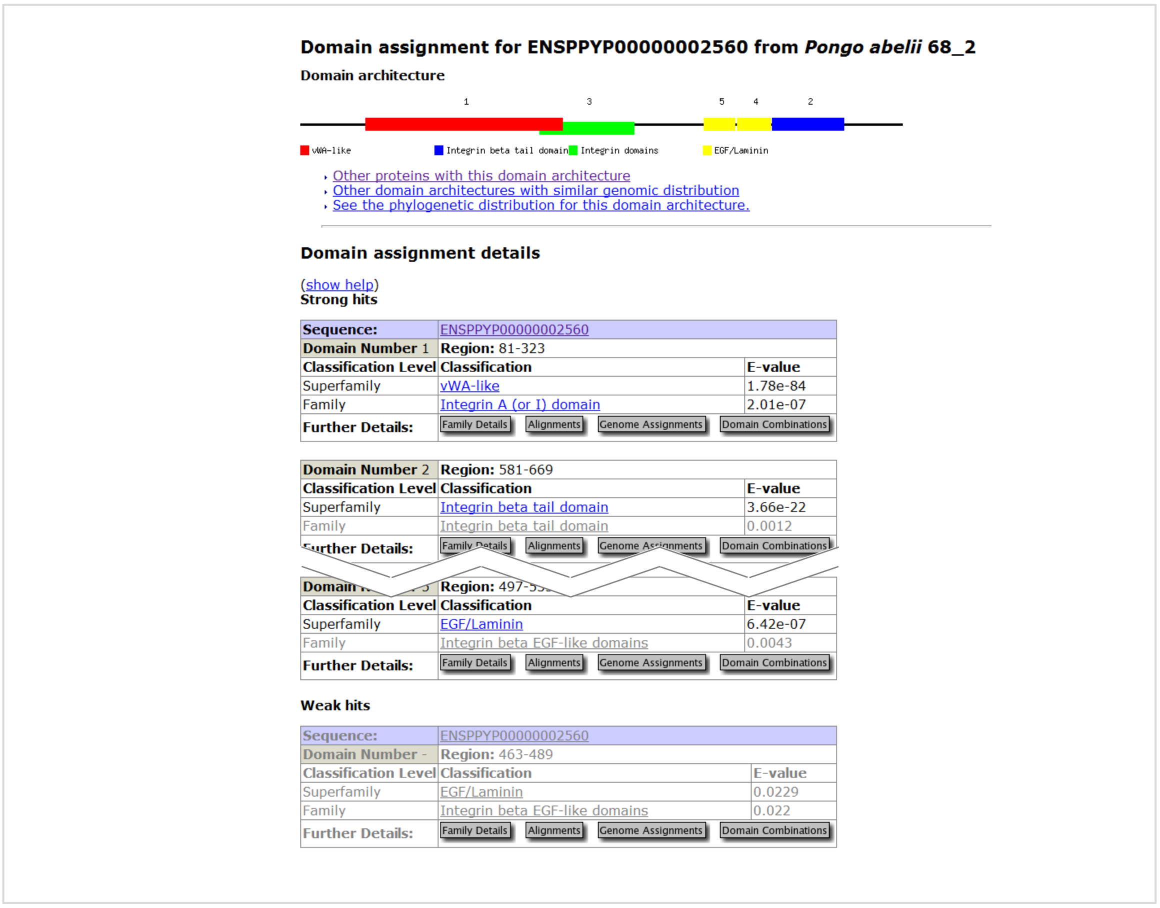
SCOP is best known for grouping protein domains at the superfamily level provided the structure, sequence and function lead to the conclusion that there is a common evolutionary ancestor. What is interesting about domains at the superfamily level is that some of their sequences have diverged greatly and share less than 35% similarity and still maintain a similar function.

### 2.1.5 SUPERFAMILY

The SUPERFAMILY database is a collection of Hidden Markov Models (HMMs) for all proteins of known structure which predicts the domain architecture of protein sequences and classifies them at the SCOP superfamily level . The database has evolved over the years and added many dimensions to its primary function and is now heavily utilized by researchers worldwide. However, for this thesis we use SUPERFAMILY to detect and classify protein domains for which there is a know protein structure based only on the protein amino acid sequence. Thus, the objective is to assign structural domains at the SCOP superfamily level.

SUPERFAMILY is used to detect and annotate domains in SCOP by using a library of HMMs that profile all superfamilies within the SCOP database. HMM profiling is similar to PSI-BLAST and both are methods used for identifying more distant relationships between protein sequences than by simply pairwise sequence comparison which will be explained further in a subsequent section.

The most popular function of the SUPERFAMILY web interface is the Sequence Search which automatically annotates SCOP (Structural Classification of Proteins) domains based on a submitted sequence. The search process starts with checking the database for already existing domain assignments. In case there are no results, then the search sequence will be compared with ASTRAL using BLAST. If domains are still not assigned then domains are searched for with the HMM library and the SAM package.



**Domain assignment for ENSPPYP00000002560 from *Pongo abelii* 68_2**

**Domain architecture**

🟥 vWA-like 🟦 Integrin beta tail domain 🟩 Integrin domains 🟨 EGF/Laminin

- Other proteins with this domain architecture
- Other domain architectures with similar genomic distribution
- See the phylogenetic distribution for this domain architecture.

**Domain assignment details**

(show help)
**Strong hits**

| **Sequence:** | ENSPPYP00000002560 | |
|---|---|---|
| **Domain Number** 1 | **Region:** 81-323 | |
| **Classification Level** | **Classification** | **E-value** |
| Superfamily | vWA-like | 1.78e-84 |
| Family | Integrin A (or I) domain | 2.01e-07 |
| **Further Details:** | Family Details Alignments Genome Assignments Domain Combinations | |

| **Domain Number** 2 | **Region:** 581-669 | |
|---|---|---|
| **Classification Level** | **Classification** | **E-value** |
| Superfamily | Integrin beta tail domain | 3.66e-22 |
| Family | Integrin beta tail domain | 0.0012 |
| **Further Details:** | Family Details Alignments Genome Assignments Domain Combinations | |

| **Domain** [illegible] | **Region:** 497-[illegible] | |
|---|---|---|
| **Classification Level** | **Classification** | **E-value** |
| Superfamily | EGF/Laminin | 6.42e-07 |
| Family | Integrin beta EGF-like domains | 0.0043 |
| **Further Details:** | Family Details Alignments Genome Assignments Domain Combinations | |

**Weak hits**

| **Sequence:** | ENSPPYP00000002560 | |
|---|---|---|
| **Domain Number** - | **Region:** 463-489 | |
| **Classification Level** | **Classification** | **E-value** |
| Superfamily | EGF/Laminin | 0.0229 |
| Family | Integrin beta EGF-like domains | 0.022 |
| **Further Details:** | Family Details Alignments Genome Assignments Domain Combinations | |

Figure 2-3: SUPERFAMILY domain assignment page for a protein sequence. Source: http://supfam.cs.bris.ac.uk

On average, the models in the HMM library generate significant assignments for more than half of the sequences (>60%). To improve the coverage of assignments for remote homologues a profile to profile HMM method can be used which works as follows. First, a few homologues of the search sequence are aligned and used to build a HMM profile. Afterwards the HMM profile is used to search the SUPERFAMILY model library for distant homologues.

Moreover, the database also contains 2,414 completely mapped genomes (started with 58 genomes in 2002!) which can be used through the comparative genomics tools to identify the over and under-represented superfamilies or families of protein domains in each genome, relative to the kingdom it belongs to (http://supfam.cs.bris.ac.uk/SUPERFAMILY/articles/nar2009.pdf).

Based on the usage of the SUPERFAMILY pipeline it is easy to understand the motivation of incorporating context for domain identification.

### 2.1.6 Conclusion

Proteins are the building blocks of life and they are formed from domains which are evolutionary units. Proteins can have one or multiple domains. Domains describe the conserved elements within proteins based on sequence or structural homology. SCOP is a database that classifies all known proteins from PDB into domains based on their similarity. SUPERFAMILY annotates protein sequences based on SCOP domains classified at the superfamily level (members that share a common evolutionary ancestor).

## 2.2 Methods for homology detection

A major role of bioinformatics is the analysis of sequence data. The purpose of protein sequence analysis is to predict the protein's function. This function is often inferred from protein homologous sequences for which the function is known. Thus, homology searches represent the predominant application of bioinformatics and a multitude of efficient searches have been developed. One of the state-of-the art such methodologies is the Hidden Markov Model (HMM), which performs a more sensitive homology search with the purpose of identifying remote homologous sequences. The remote homologues search is gaining territory and new or improved tools to search for them have been developed.

### 2.2.1 Homologous and remote homologous proteins

The problem that bioinformaticians are now facing is annotating new protein sequences with both structural and functional features. At the moment, this can be done via detection of protein homology. Protein homology basically means that similar proteins originate from the same ancestor. Protein homology detection is a major topic in bioinformatics and is used to classify protein domains, in our case at the SCOP superfamily level. There have been developed many statistical tools for detecting protein homologies and the simplest one is through sequence similarity.

*Sequence alignments* is a method of arranging the amino-acid sequence of proteins in order to determine similar regions within the sequence that may be caused by structural, functional or evolutionary relationships within the protein sequences.

The most common way of identifying homologous proteins is to perform a Multiple Sequence Alignment (MSA), where the protein sequences are compared to one another at the amino acid level. In order to inspect the evolutionary relationship among these proteins, gaps are introduced wherever an insertion or deletion occurred. Essentially, a MSA is a hypothetical map which presents the mutations that have arisen during a given sequence's evolution.

Detecting homologies when there is a high level of sequence similarity can be accurately performed with the current bioinformatics tools. The challenge follows when trying to detect homologies at low levels of sequence similarity (remote homology detection). These remote homologous proteins still share a common ancestor even if at structural level they have diverged considerably from the primary protein sequence. When performing multiple sequence alignment on a protein sequence with the purpose of identifying the available homologues two issues may arise: either the sequence similarity is below 30%

which makes it difficult to decide on the homologous status of a candidate protein ([1]) or the proteins in question may show conservation sites, but structurally they have nothing in common. This is why remote homolog detection is complex as no scientist can make inferences solely based on structural analysis or on sequence analysis.

In an attempt to avoid the above mentioned problems, many powerful algorithms were developed. These methods can be divided into three main categories: methods that search for sequence similarity; generative methods where a probabilistic model (like a Hidden Markov Model) is built and a candidate protein is compared to see the goodness of fit; and lastly discriminative methods where homologous proteins are compared to non-homologous ones and these attributes are then used to determine whether a candidate protein is homologous or not ([2]). These methods will be further discussed in the following sections.

### 2.2.2 Sequence Similarity Search

Sequence similarity searching is one of the most common bioinformatics tools used for inferring homologues from a query protein. Through this method a series of sequence databases is searched by using alignments to a candidate protein. Two of the most common approaches of this method are BLAST ([11]) and FASTA ([12]). Contrary to dynamic programming which indeed gives the best alignment between two sequences but at the expense of processing time, BLAST and FASTA base their search on heuristics. These algorithms prune the search tree using heuristics in selecting only those sequences available that are more likely to be similar to the query protein. To go more in-depth, first, fixed-length quotes are extracted from the query sequence and then the algorithm searches the entire database for these exact matches. After this first round of results, the top hits are extended for longer similar quotes. This heuristic approach is usually 50 to 100 times faster than dynamic programming.

In order to assess the significance of the sequence alignment, a scoring system is put in place. This is based on the scoring substitution matrix, like BLOSUM62 or PAM and gap penalties. Substitution matrixes give a score to each substitution of one residue for another, while the gap penalty mimics the evolutionary process that leads to the formation of these gaps. With this scoring system, a raw score of the alignment can be computed as the sum of the amino acid substitution scores and the gap penalties. However, this score is not enough for finding the optimal homologues of a protein; in this case a statistical assessment of the alignment is required. BLAST and FASTA verify if an alignment is biologically relevant with the use of an Expected Value or E-value. This E-value denotes the number of distinct alignments with an equal or higher score than that of the query sequence one could expect to find by chance in a database of a certain size; this number decreases exponentially as the score of the found match increases. In fact, the closer the E-value is to the zero mark, the more significant is the protein match. Nevertheless, the expected value is not enough to incur statistical significance since it is computed with the length of the query sequence in mind making shorter sequences to have a higher probability of occurring in a given database by chance ([13]). Next to the expected value notion we have to introduce the P-value, which is the probability of obtaining by chance a score at least as large as that of the query sequence. In fact, the E-value is the product of the P-value and the size of the database.

Sequence similarity search methods are suitable for detecting homologues only when protein sequences larger than 100 residues in length have 30% identical amino acids with the query sequence. Anything below this value falls within the Twilight Zone making remote homology detection extremely inefficient.

### 2.2.3 Generative methods

BLAST and FASTA proved to be insufficient for remote homology detection and thus, later approaches were constructed. Among these profile and profile Hidden Markov Models must be mentioned; the algorithm relies on aggregating statistics from a group of protein sequences belonging to the same family or superfamily into a probabilistic model which is then used to test the similarity of the query sequence against the model.

First, the Position-Specific Iterative BLAST (PSI-BLAST) will be presented, then the profile Hidden Markov Models (pHMM) or sequence-profile oriented approaches and lastly the introduction of domain co-occurrence or domain architecture information into improving these methods will be discussed.

#### PSI-BLAST

Position-Specific Iterative BLAST is a computational tool that derives a position- specific scoring matrix (PSSM) or profile from the multiple sequence alignment of sequences detected above a certain threshold score using protein-protein BLAST with the purpose of identifying distant evolutionary relationships among proteins ([14]).

The PSI-BLAST search is a repetitive process with the first iteration being identical to a run of the BLASTp (BLAST specific only to protein sequences, hence the p at the end) algorithm. The next step is to aggregate the multiple alignment sequences with the highest E-values above the preset threshold and then combine them in a probabilistic model or a Position-specific scoring matrix. This matrix stores the conservation patterns captured from the MSA in the form of a score matrix; highly conserved positions are rewarded high scores, whilst low conservation areas receive almost zero scores. The next step is to substitute the BLOSUM62 matrix that BLASTp uses for this PSSM and run a new search of the database for similar protein sequences now based on the new PSSM conservation patterns. The new matches are added to the previously created profile and a new scoring matrix is computed. This process is continued until convergence, i.e. until no more protein sequences can be found above the preset E-value threshold.

So far, PSI-BLAST has proved to be a better search tool than Sequence Similarity Search algorithms for detecting remote homologues; since the former utilizes the conservation information retrieved from the multiple alignments it can identify distant homologues similar to a 3D structure search.

#### Profile Hidden Markov Models

A Hidden Markov Model is a statistical model that constructs a profile of the query sequence given the multiple sequence alignments of homologous sequences. In fact, a profile HMM is a linear series of nodes that roughly correspond to the residue position from the alignment it was constructed (see **Error! Reference source not found.**). Each ode can have one of the following states that comprise the rate of mutation:

- Match (M) = this state models the conserved regions within the alignment;
- Insert (I) = portions of sequences that do not match within the original model;
- Delete (D)

Insertions and deletions states actually model indel states (mutation classes that include both deletions and insertions).

Two types of probabilities are associated with a profile Hidden Markov Model: the transition probability of transition from one state to another; and the emissions probability which is associated with the each match state based on the probability of a given amino acid to exist at that same position within the alignment. Concerning the emission state, deletions are silent leaving each match and insert state with an emission probability distribution.

Profile HMMs can be used as representations of sequence families or domains making them ideal for identifying distant sequence homologues. The performance of these models is highly dependent on the quality of the estimated transition and emission probabilities. The quality is usually poor because when searching for remote homologues, the global alignment will have too few sequences to estimate the above mentioned probabilities with confidence. One solution is the use of Dirichlet mixtures, which are methods for estimating probabilities of amino acids within small sample size. Moreover, similar sequences carry less weight since the residues are conserved with a higher weight than when dealing with distant/divergent sequences ([2]).

After the pHMM has been built, the profile can be used to align and score a query protein sequence with the newly created model. For this, dynamic programming is employed, and more specifically the Viterbi algorithm which finds the most likely path to follow a sequence of events ([15]). The algorithm computes the probability that the query sequence $S_n$ will match the profile HMM *M(w),* where n is the length of the protein sequence and w the set of estimated parameters for the probabilistic model:

$$P\big(S_n \big| M(w)\big) = \sum\nolimits_{\pi} P\big(S_n, \pi \big| M(w)\big),$$

Where $\pi$ is the pathway of the pHMM that starts at the Begin state and ends at the End state.

The algorithm aims to solve $\pi^* = argmax_{\pi}\,(x, \pi | M(w))$.

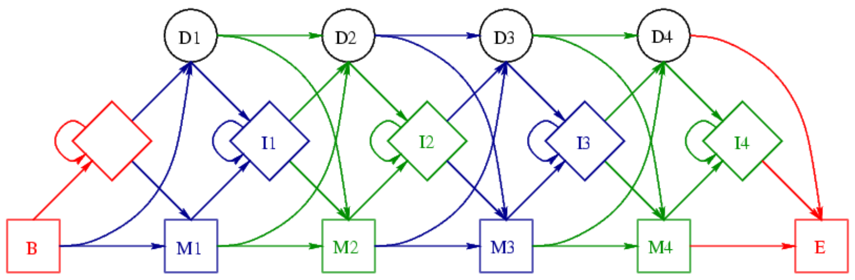
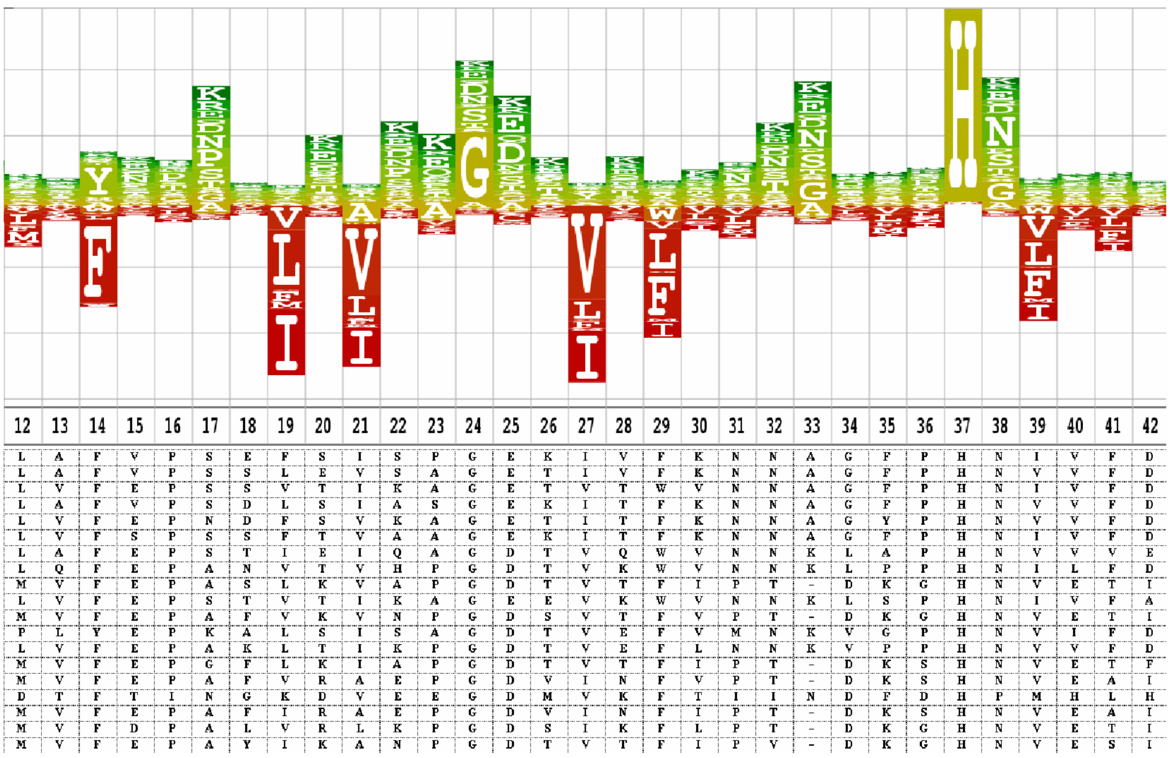


Figure 2-4 Hidden Markov Model States (Source: Julian Gough, Bristol University, Computational Genomics and Bioinformatics Algorithms, Lecture Notes, 2012)



| 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| L | A | F | V | P | S | E | F | S | I | S | P | G | E | K | I | V | F | K | N | N | A | G | F | P | H | N | I | V | F | D |
| L | A | F | V | P | S | S | L | E | V | S | A | G | E | T | I | V | F | K | N | N | A | G | F | P | H | N | V | V | F | D |
| L | V | F | E | P | S | S | V | T | I | K | A | G | E | T | V | T | W | V | N | N | A | G | F | P | H | N | I | V | F | D |
| L | A | F | V | P | S | D | L | S | I | A | S | G | E | K | I | T | F | K | N | N | A | G | F | P | H | N | V | V | F | D |
| L | V | F | E | P | N | D | F | S | V | K | A | G | E | T | I | T | F | K | N | N | A | G | Y | P | H | N | V | V | F | D |
| L | V | F | S | P | S | S | F | T | V | A | A | G | E | K | I | T | F | K | N | N | A | G | F | P | H | N | I | V | F | D |
| L | A | F | E | P | S | T | I | E | I | Q | A | G | D | T | V | Q | W | V | N | N | K | L | A | P | H | N | V | V | V | E |
| L | Q | F | E | P | A | N | V | T | V | H | P | G | D | T | V | K | W | V | N | N | K | L | P | P | H | N | I | L | F | D |
| M | V | F | E | P | A | S | L | K | V | A | P | G | D | T | V | T | F | I | P | T | – | D | K | G | H | N | V | E | T | I |
| L | V | F | E | P | S | T | V | T | I | K | A | G | E | E | V | K | W | V | N | N | K | L | S | P | H | N | I | V | F | A |
| M | V | F | E | P | A | F | V | K | V | N | P | G | D | S | V | T | F | V | P | T | – | D | K | G | H | N | V | E | T | I |
| P | L | Y | E | P | K | A | L | S | I | S | A | G | D | T | V | E | F | V | M | N | K | V | G | P | H | N | V | I | F | D |
| L | V | F | E | P | A | K | L | T | I | K | P | G | D | T | V | E | F | L | N | N | K | V | P | P | H | N | V | V | F | D |
| M | V | F | E | P | G | F | L | K | I | A | P | G | D | T | V | T | F | I | P | T | – | D | K | S | H | N | V | E | T | F |
| M | V | F | E | P | A | F | V | R | A | E | P | G | D | V | I | N | F | V | P | T | – | D | K | S | H | N | V | E | A | I |
| D | T | F | T | I | N | G | K | D | V | E | E | G | D | M | V | K | F | T | I | I | N | D | F | D | H | P | M | H | L | H |
| M | V | F | E | P | A | F | I | R | A | E | P | G | D | V | I | N | F | I | P | T | – | D | K | S | H | N | V | E | A | I |
| M | V | F | D | P | A | L | V | R | L | K | P | G | D | S | I | K | F | L | P | T | – | D | K | G | H | N | V | E | T | I |
| M | V | F | E | P | A | Y | I | K | A | N | P | G | D | T | V | T | F | I | P | V | – | D | K | G | H | N | V | E | S | I |

Figure 2-5 Profile Hidden Markov Model (Source: http://supfam.org)

### 2.2.4 Discriminative methods

Two of the discriminative methods that have arisen as an alternative to the profile HMM described under generative methods are Support Vector Machines (SVM) ([16]) and Inductive Logic Programming (ILP) ([17]).

SVMs are trained on a specific classification function in order to discriminate members of a given protein (super)family from non-members. The features that are incorporated in the SVM are obtained either through comparisons of the query sequence to the profile HMM built from the training dataset or by use of primary sequence motifs ([2]). Although SVMs are outperforming the generative methods, they are still heavily reliant on them for feature extractions. Likewise, the computational time needed might be too costly when adopting these machine learning techniques.

As for the Inductive Logic Programming approach to remote homology detection, researchers have developed two methods. The first one is called Homology Induction (HI), whilst the second one uses a combination of machine learning and ILP to directly functional protein classes from sequences. HI learns rules that are true for sequences that are highly similar to the query one and false rules for general or non-homologous sequences. The list of sequences is obtained after running PSI-BLAST on the query sequence. Even though, at first glance, HI would outperform the previously described homology detection methods it must be mentioned that the quality of prediction is heavily reliant on the list of homologues derived from PSI-BLAST. This makes HI highly susceptible to the performance of generative methods in the first place.

The second IPL method creates first-order logic predicates out of all the homologous sequences. The predicates are constructed directly from the properties that the protein sequences exhibit. The next step is to run an IPL data mining algorithm that will find the most frequent patterns found among the homologues. After this, the patterns will be transformed into binary attributes that will later be used to train a machine learning algorithm, i.e. decision trees in this case.

### 2.2.5 Conclusion

Both sequence similarity searching and generative methods are wildly used for homology detection and annotation of proteins and their domains. This is mostly because of their computational time, which remains the biggest drawback of the discriminative methods that use SVMs and IPL.

## 2.3 **Mathematics of scoring**

Both the sequence similarity and the generative methods described earlier depend on a scoring scheme, which is a function that ranks the alignments based on the degree of similarity of the sequences. The scheme that we used in our thesis is additive and is derived from on [10].

### 2.3.1 Substitution matrices

Let us assume that we have two protein sequences in amino acid form:

$$x = x_1 \dots x_n$$

$$y = y_1 \cdots y_m$$

where $x_i, y_i \in \mathcal{A} = \{\mathrm{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V}\}$ are the amino acid positions which are selected from the alphabet $\mathcal{A}$.

If we already have a pair of sequences which are aligned, we want to assign a score to the alignment which would give a measure of the likelihood that the sequences are related versus unrelated. We achieve this by calculating two probabilities for the random model $R$ and for the match model $M$.

The random model assumes that the amino acid $a$ occurs with the frequency $q_a$, therefore the probability of having the two sequences is the product of the probabilities of each amino acid:

$$\text{random model: } P(x,y|R) = \prod_i q_{x_i} \prod_j q_{y_j}$$

In the case of the match model the aligned pairs of amino acids occur with the joint probability $p_{ab}$. The joint probability $p_{ab}$ is the probability that amino acids $a$ and $b$ have each independently been derived from some original amino acid $c$ in their common ancestor. The probability of the whole alignment is:

$$\text{match model: } P(x,y|M) = \prod_i p_{x_i y_i}$$

To calculate the odds ratio we divide the likelihood of the match model to the likelihood of the random model.

$$odds\ ratio: \frac{P(x,y|M)}{P(x,y|R)} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}}$$

We then calculate the log odds ratio

$$S = \sum_i s(x_i, y_i)$$

where

$$s(a,b) = \log \frac{p_{ab}}{q_a q_b}$$

The $s(a_i, a_j)$ scores can be placed in a 20x20 matrix for all amino acid combinations. It is called a score matrix or a substitution matrix, and we give as an example below the BLOSUM50 ([10]) matrix where the element $i, j$ is the score for amino acids $a_i$ and $a_j$. The log-odds values have been scaled and rounded for ease of computation. It is evident that the values on the main diagonal for identical amino acids are the highest.

| | A | R | N | D | C | Q | E | G | H | I | L | K | M | F | P | S | T | W | Y | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 5 | -2 | -1 | -2 | -1 | -1 | -1 | 0 | -2 | -1 | -2 | -1 | -1 | -3 | -1 | 1 | 0 | -3 | -2 | 0 |
| R | -2 | 7 | -1 | -2 | -4 | 1 | 0 | -3 | 0 | -4 | -3 | 3 | -2 | -3 | -3 | -1 | -1 | -3 | -1 | -3 |
| N | -1 | -1 | 7 | 2 | -2 | 0 | 0 | 0 | 1 | -3 | -4 | 0 | -2 | -4 | -2 | 1 | 0 | -4 | -2 | -3 |
| D | -2 | -2 | 2 | 8 | -4 | 0 | 2 | -1 | -1 | -4 | -4 | -1 | -4 | -5 | -1 | 0 | -1 | -5 | -3 | -4 |
| C | -1 | -4 | -2 | -4 | 13 | -3 | -3 | -3 | -3 | -2 | -2 | -3 | -2 | -2 | -4 | -1 | -1 | -5 | -3 | -1 |
| Q | -1 | 1 | 0 | 0 | -3 | 7 | 2 | -2 | 1 | -3 | -2 | 2 | 0 | -4 | -1 | 0 | -1 | -1 | -1 | -3 |
| E | -1 | 0 | 0 | 2 | -3 | 2 | 6 | -3 | 0 | -4 | -3 | 1 | -2 | -3 | -1 | -1 | -1 | -3 | -2 | -3 |
| G | 0 | -3 | 0 | -1 | -3 | -2 | -3 | 8 | -2 | -4 | -4 | -2 | -3 | -4 | -2 | 0 | -2 | -3 | -3 | -4 |
| H | -2 | 0 | 1 | -1 | -3 | 1 | 0 | -2 | 10 | -4 | -3 | 0 | -1 | -1 | -2 | -1 | -2 | -3 | 2 | -4 |
| I | -1 | -4 | -3 | -4 | -2 | -3 | -4 | -4 | -4 | 5 | 2 | -3 | 2 | 0 | -3 | -3 | -1 | -3 | -1 | 4 |
| L | -2 | -3 | -4 | -4 | -2 | -2 | -3 | -4 | -3 | 2 | 5 | -3 | 3 | 1 | -4 | -3 | -1 | -2 | -1 | 1 |

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| K | -1 | 3 | 0 | -1 | -3 | 2 | 1 | -2 | 0 | -3 | -3 | 6 | -2 | -4 | -1 | 0 | -1 | -3 | -2 | -3 |
| M | -1 | -2 | -2 | -4 | -2 | 0 | -2 | -3 | -1 | 2 | 3 | -2 | 7 | 0 | -3 | -2 | -1 | -1 | 0 | 1 |
| F | -3 | -3 | -4 | -5 | -2 | -4 | -3 | -4 | -1 | 0 | 1 | -4 | 0 | 8 | -4 | -3 | -2 | 1 | 4 | -1 |
| P | -1 | -3 | -2 | -1 | -4 | -1 | -1 | -2 | -2 | -3 | -4 | -1 | -3 | -4 | 10 | -1 | -1 | -4 | -3 | -3 |
| S | 1 | -1 | 1 | 0 | -1 | 0 | -1 | 0 | -1 | -3 | -3 | 0 | -2 | -3 | -1 | 5 | 2 | -4 | -2 | -2 |
| T | 0 | -1 | 0 | -1 | -1 | -1 | -1 | -2 | -2 | -1 | -1 | -1 | -1 | -2 | -1 | 2 | 5 | -3 | -2 | 0 |
| W | -3 | -3 | -4 | -5 | -5 | -1 | -3 | -3 | -3 | -3 | -2 | -3 | -1 | 1 | -4 | -4 | -4 | 15 | 2 | -3 |
| Y | -2 | -1 | -2 | -3 | -3 | -1 | -2 | -3 | 2 | -1 | -1 | -2 | 0 | 4 | -3 | -2 | -2 | 2 | 8 | -1 |
| V | 0 | -3 | -3 | -4 | -1 | -3 | -3 | -4 | -4 | 4 | 1 | -3 | 1 | -1 | -3 | -2 | 0 | -3 | -1 | 5 |

Table 2-1 BLOSUM50 substitution matrix (Source: [10])

### 2.3.2 Significance of scores

To measure the significance we can check the distribution of the maximum of $N$ alignment scores to independent random sequences. The score is considered significant if the probability of this maximum being greater than the observed best score is small. Let $M_N$ be the maximum of a series of N independent Gaussian random variables. Then asymptotically we converge to the extreme value distribution (EVD):

$$P(M_N \leq x) \cong exp\left(-KNe^{\lambda(x-\mu)}\right)$$

The number of unrelated matches with score greater than $S$ is approximately Poisson distributed with mean:

$$E(S) = Kmne^{-\lambda S}$$

where

- $\lambda$ is the positive root of $\sum_{a,b} q_a q_b e^{\lambda s(a,b)} = 1$;
- $K$ is a constant given by a geometrically convergent series also dependent on the $q_a$ and $s(a,b)$; $K$ corrects for the non-independence of possible starting points for matches;
- $\lambda$ is a parameter to transform $s(a,b)$ into a natural scale.

The probability that there is a match of score greater than $S$ is then:

$$P(x > S) = 1 - e^{-E(S)}$$

### 2.3.3 The Extreme Value Distribution

Let $g(x)$ be the probability density function and we take $N$ samples. The probability that the largest of $N$ samples is less than $x$ is $G^N(x)$, where $G(x) = \int_{-\infty}^{x} g(u)du$. Then the probability density function of $G^N(x)$ is it's derivative with respect to $x$:

$$\frac{\partial G^N(x)}{\partial x} = Ng(x)G^{N-1}(x) \rightarrow extreme\ value\ density\ (EVD)\ for\ g(x)$$

If we are dealing with the exponential density: $g(x) = \alpha e^{-\alpha x}$ then the probability distribution function is $G(x) = 1 - e^{-\alpha x}$. If we choose $y$ such that $e^{-\alpha y} = 1/N$ and define $z = x - y$, then:

$$Ng(x)G^{N-1}(x) = N\alpha e^{-\alpha x}(1 - e^{-\alpha x})^{N-1} = \alpha e^{-\alpha z}(1 - e^{-\alpha z}/N)^{N-1}$$

$$\rightarrow \alpha e^{-\alpha z} \exp(-e^{-\alpha z}) \ for \ N \rightarrow \infty$$

Therefore if we calculate the integral

$$\int_{-\infty}^{x} \alpha e^{-\alpha z} \exp(-e^{-\alpha z})\, dz = \exp(-e^{-\alpha x})$$

we get the probability distribution function:

$$P(extreme\ value \leq x) = \exp(-e^{-\alpha x})$$

which is called a Gumbel distribution.

The Gumbel distribution is the EVD for various other densities $g(x)$, for example the Gaussian.

## 2.4 **Previous Work on Context Sensitive Domain Prediction**

Current domain assignment models try to predict each domain independently. Nevertheless, most proteins contain multiple domains which appear in very specific combinations. This leads to the natural assumption that one domain prediction can influence the adjacent domain predictions.

The first paper to use context for domain prediction in protein sequences of amino acids was [18]. The authors show that statistical language modelling techniques applied to protein sequences can enhance domain recognition significantly. They try two types of Hidden Markov Models which use the dependence between domains in a sequence. The first version of the model has a small number of parameters but has a limited prediction power. The second version has higher prediction power but the number of parameters increases exponentially, leading to difficulties in estimation.

The second paper to use context was [19], where the authors built the AIDAN tool (Automated Improvement of Domain ANnotations). This tool clusters domain sequences which are similar and based on these clusters it predicts whether there are any domain assignments which may be mistakes. As this method uses protein domain architectures as context, it provides further evidence that building a context sensitive domain prediction tool is worthwhile. However, this approach is complementary to our proposed research and can be used in conjunction with it.

Paper [20] describes another context sensitive approach for improving the precision of HMM domain detection. The CODD algorithm functions along the following lines:

- First build a reference list of conditionally dependent pairs (CDP), which contains all the domain pairs that co-occur at least once in the set of proteins of known composition;
- Then use a HMM model to predict domains on the query sequence with a permissive E-value;
- If one of these predicted domains forms with another non-overlapping predicted domain a pair that is found in the CDP list, then the two domains are considered to be certified;
- The certified domains are then used to decide on the final predictions.

This approach is less flexible than [21] as it only uses a favoured domain pairs list in which all co-occurrences are treated with the same weight. Other negative sides are firstly that the order of the co-occurring domains is not taken into account, and secondly that the evaluation of the method is done using only one species. Nevertheless, this approach was a precursor for paper [21].

Paper [21] uses protein domain co-occurrence statistics to improve individual domain predictions. It can be valuable to our project as it proves that the use of protein domain architecture information can have a significant impact on prediction quality for a wide range of organisms. The authors use a graph-theoretic framework to find the set of domains that maximize an overall score. The method can boost up the accuracy of standard procedures with up to 11%, without significant additional computing cost. This approach will be central to my project as it is more accurate than the approach of [20]. The next section presents the model from paper [21] in more detail.

### 2.4.1 The dPUC model

This section explains the details of the dPUC model from [21]. Let’s assume we have a protein sequence and $P$ is the set of candidate domains obtained through HMMER with a tolerant E-value threshold. Now let’s assume $D$ is a subset of $P$. The authors define the score of domain $i$ with respect to $D$ as:

$$S_{i,D} = H_i - T_i + \sum_{j \in D} C_{ij}$$

where

- $H_i$ is the score of domain $i$, which is calculated in a similar way as in section 2.3
- $T_i$ is the domain score threshold for the family of domain $i$, and
- $C_{ij}$ is the context score between domains $i$ and $j$.

The calculation of $H_i$ and $T_i$ is already included in the existing standard scoring frameworks. So the novelty of this model is the introduction of the context score $C_{ij}$ which fits well with an additive scoring scheme.

The next few paragraphs are dedicated to the context scores calculation. For calculating $C_{ij}$ we need statistics from an already annotated protein domain architecture database. The authors of the dPUC model used Pfam domain assignments based on sequences for about 10 organisms from Uniprot. In our project we will use SUPERFAMILY domain assignments based on sequences for more organisms from the Protein Data Bank. The authors used only multi-domain proteins which is a good idea because a single domain protein cannot provide context statistics. We first define the *normalized pair counts* $c_{ij}$ as the sum over all proteins $p$ in the database of the product of the counts of domain $i$ and $j$ normalized by the total number of domains in protein $p$:

$$c_{ij} = \sum\nolimits_p \frac{e_{ip}e_{jp}}{e_p - 1}, \qquad \forall i,j \; if \; i \neq j$$

$$c_{ii} = \sum\nolimits_p \frac{e_{ip}(e_{ip} - 1)}{e_p - 1}, \qquad \forall i$$

where $e_{ip}$ is the number of domains of family $i$ in protein $p$ and $e_p$ the total number of domains in protein $p$. The normalization done with $e_p - 1$ is meant to compensate for inflated counts due to very large proteins. Also in the case of $i = j$ the expression $e_{ip}(e_{ip} - 1)$ keeps domains from counting themselves in the case where we only have one occurrence of family $i$ in protein $p$. Having $c_{ij}$ defined, we can now calculate probabilities:

$$p_{ij} = \frac{c_{ij} + \frac{1}{n^2}}{c + 1}, \qquad \forall i,j$$

$$p_i = \frac{1}{n}, \qquad \forall i,$$

where $p_{ij}$ is the probability that domains $i$ and $j$ co-ocurred, $p_i$ is the probability that a random domain family occurred, $c$ is the sum of all $c_{ij}$ ($c = \sum_{ij} c_{ij}$) and $n$ is the total number of unique domain families. The probabilities are scaled such that $\sum_{ij} p_{ij} = 1$ and $\sum_i p_i = 1$.

We now have the building blocks to calculate the context scores $C_{ij}$ as pairwise log-odds:

$$C_{ij} = \log_2 \frac{p_{ij}}{p_i p_j}, \qquad \forall i,j,$$

where $p_{ij}$ is the measured probability that domains $i$ and $j$ co-ocurred, and $p_i p_j$ is the probability that the association between domains $i$ and $j$ occurred by chance. The logarithm in base two is chosen such that the context scores match the HMMER2 bit scores.

What is useful about this formulation is that if the domain pair $i,j$ is not observed, the context score is negative. But if the pair $i,j$ is observed and the context score is still negative, then it is automatically set to zero so it will have no influence on the total score.

These statements are evident from the equations above. For example if the domain pair $i, j$ is not observed, the context score is negative: $c_{ij} = 0 \rightarrow C_{ij} = \log_2 \frac{1}{1+c}$<0.

Let’s come back to our final aim of selecting a set of domain assignments. The objective is to find $D = argmax(\sum_{i \in D} S_{i,D})$, where $D$ only contains non-overlapping domains and $\forall i \in D, S_{i,D} \geq 0$.

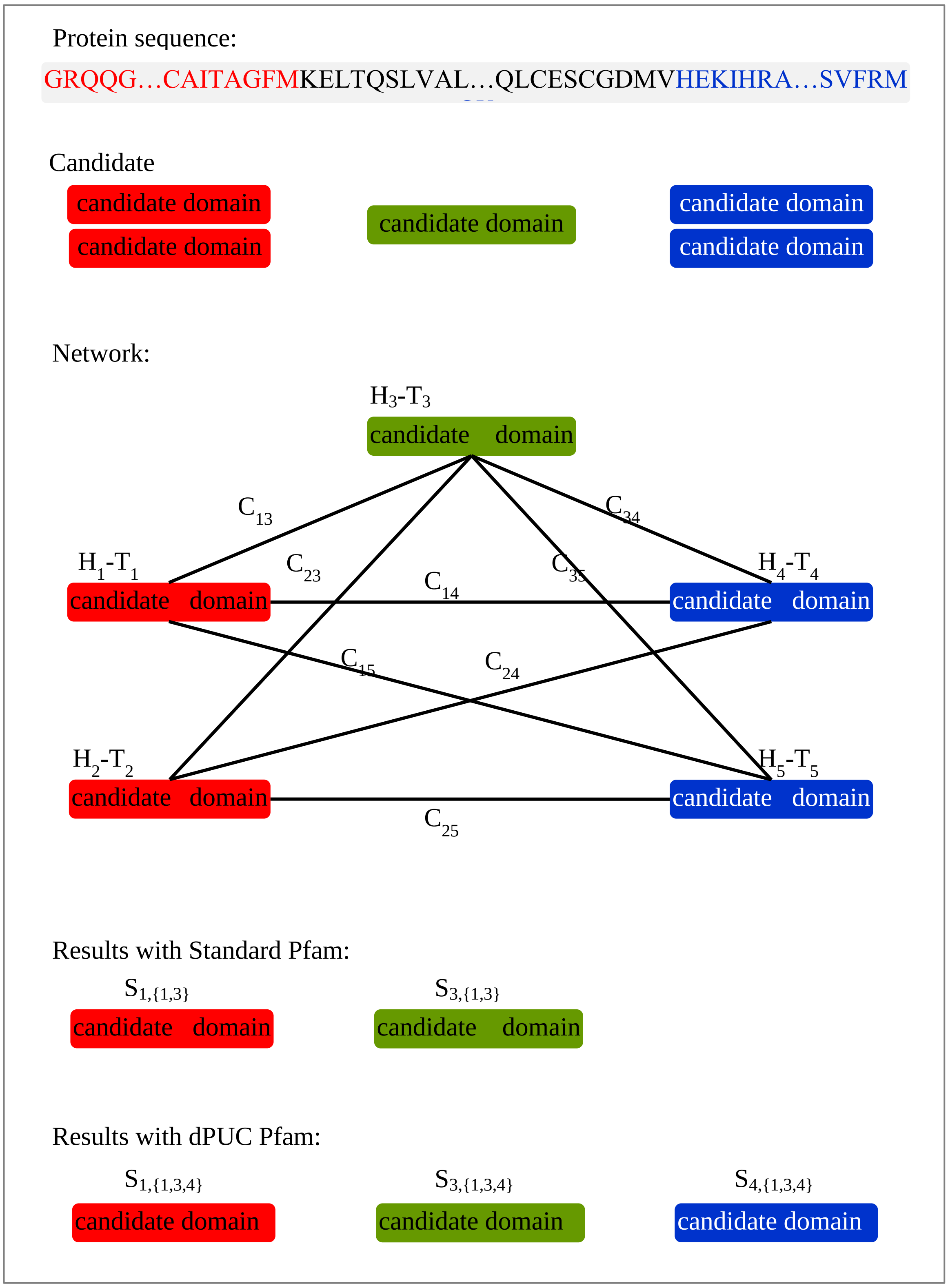


Figure 2-6 Graph theoretic framework

This problem can be modelled from a graph theory perspective. If we look at Figure 2-6 we see at the upper part an example of protein sequence in amino acid form. In this example this protein sequence can be split into three parts each of them having different candidate domain assignments discovered by HMMER with a permissive threshold. For the first domain position there are two candidate domains (1 and 2), for the second there is one candidate domain (3) and for the last position there are two candidate domains (4 and 5). From these candidate domains we build a network in which each node has the weight equal to the Pfam score minus the threshold $H_i - T_i$ and the branches have the weights equal to the context scores $C_{ij}$. In the bottom of the figure you can see the final non-overlapping domain assignment results for the Standard Pfam model and for the Pfam with dPuc. The standard model has fewer results (two) than dPUC which has three results due to positive context for domain 4.

To find the solution to this optimization task, the authors of [21] formulate it as an integer linear programming problem. For this purpose, we define a variable $x_i \in \{0,1\}$ for each domain $i$ and a variable $x_{ij} \in \{0,1\}$ for domains $ij$ and

$$x_i = \begin{cases} 1\ if\ domain\ i\ is\ included\ in\ the\ final\ set\ of\ predictions \\ 0\ otherwise \end{cases}$$

$$x_{ij} = \begin{cases} 1\ if\ domain\ pair\ ij\ is\ included\ in\ the\ final\ set\ of\ predictions \\ 0\ otherwise \end{cases}$$

Next we define the score of domain $i$ as:

$$S_i = [H_i - T_i]x_i + \sum_j C_{ij}x_{ij}$$

Then the goal is to maximize $\sum_i S_i$ such that:

$$\begin{cases} x_i,\ x_j,\ x_{ij} \in \{0,1\} & \forall i, j, \\ 0 \le x_i + x_j - 2x_{ij} \le 1 & \forall i, j, \\ x_i + x_j \le 1 & \forall i, j, with\ disallowed\ overlaps \\ S_i \ge 0 & \forall i\ (enforces\ domain\ threshold) \\ \sum_{i \in F} S_i + T_i x_i \ge T_F^* & \forall families\ F (enforces\ a\ Pfam\ threshold) \end{cases}$$

where $T_F^*$ is a Pfam-specific sequence threshold. They use lp_solve for finding the solution to this integer linear programming problem.

### 2.4.2 Evaluation of dPUC

Two methods of evaluating the performance of the model are used in [21]: Estimated false discovery rate (FDR) and Ortholog coherence scores.

We discuss here the Estimated FDR. They estimate the FDR, by comparing the standard Pfam model with the dPUC model on actual and permutated protein sequences. To simulate the false predictions they use the shuffled sequences, while the true predictions are done on the real protein sequences. Shuffling is done to preserve the fractions of aminoacids and the total length of each protein. To evaluate the context models they also concatenate the real sequences with the randomized sequences so that they can count the random predictions generated by context with real but also shuffled sequences.

To calculate the false discovery rate they compute:

$$FDR = \frac{A}{R}$$

where $A$ is the number of predictions on the shuffled protein and $R$ is the number of predictions per real protein.

For evaluating the results the authors of [21] used eight organisms: human, D. melanogaster, C. elegans, S. cerevisiae, E. coli, P. falciparum, P. vivax and M. tuberculosis. The method is tested for each of these organisms separately due to the diversity of their sequence statistics. They tested dPUC by varying the HMMER2 E-value threshold and they discovered that dPUC consistently enhances the performance of Pfam for all the organisms tested.

## 2.5 **Critical Assessment of previous research**

We have given an overview of the papers that used protein domain context for domain prediction. All of them claimed that they improved the domain detection for remote homologues. Therefore the concept of using context is already proven to work by several researchers, but we want to improve it and apply it to the SUPERFAMILY analysis pipeline and the Protein Data Bank. Nevertheless, there is one paper ([21]) which provided the best results so far and we used it as a starting point in our project.

In the next paragraphs we will assess the research done in [21] and compare it to our proposed project. The solution provided by the authors is very elegant, keeping the context score in an additive scoring framework. Compared to other methods, dPUC induces not only positive but also negative context which can allow domain assignment to be upgraded or downgraded. Our proposed scoring scheme, stayed in the additive framework and included ordering and adjacency information, in contrast to using just domain co-occurrence.

The graph theoretical approach used to find the domain assignment set is straightforward and gives reasonably fast solutions.

The accuracy boost of up to 11% compared to the standard Pfam model is quite a significant achievement.

The disadvantages of dPUC are the following. First, the authors test on few organisms. Second, the co-occurrence information can be improved by adding ordering and adjacency information in the score function.

### 2.5.1 Adjacency and ordering

There are several papers which prove that protein domain adjacency and ordering are important and should be included in the context information.

The authors of [8] show that if we find an adjacent pair of domains $A$ and $B$ then they occur in the order $AB$ or $BA$ but only in about 2% of cases they occur in both orders. In [9] it is shown that the combinations between protein domains are quite limited, most domains being observed in combination with at most two other families. At the same time 25% of the studied domains do not make combinations with other families.

There are some combinations of domains which are very frequent in proteins. In [3] these combinations are called supra-domains. The authors find 1203 two-domain and 166 three-domain combinations that are significantly over-represented.

# 3 Method

In this chapter we start presenting our method in section 3.1 which motivates our choice for adjacency and ordering and gives a high level description of our approach. We continue with section 3.2 which gives a detailed mathematical description of our context scoring frameworks and section 3.3 which presents a mathematical derivation for our E-value context factor based on context scores. Next we present the data including database schema in section 3.4. Finally we explain the benchmark concept in subsection 3.5.1 and the way we plan to test the benchmark in subsection 3.5.2.

## 3.1 Motivation for Approach

Protein domains appear in nature in only a few combinations next to each other. Out of all the possible domain combinations, less than 1% appear at least once in the UniProt database. The histogram in Figure 3-1 shows that out of the possible $2223 \cdot 2223 \cong 5\,million$ domain combinations only a few thousand occur at least once in the UniProt database. This gives strong support to our hypothesis that domain adjacency can be used as context information as only a few domain combinations are actually found in nature.

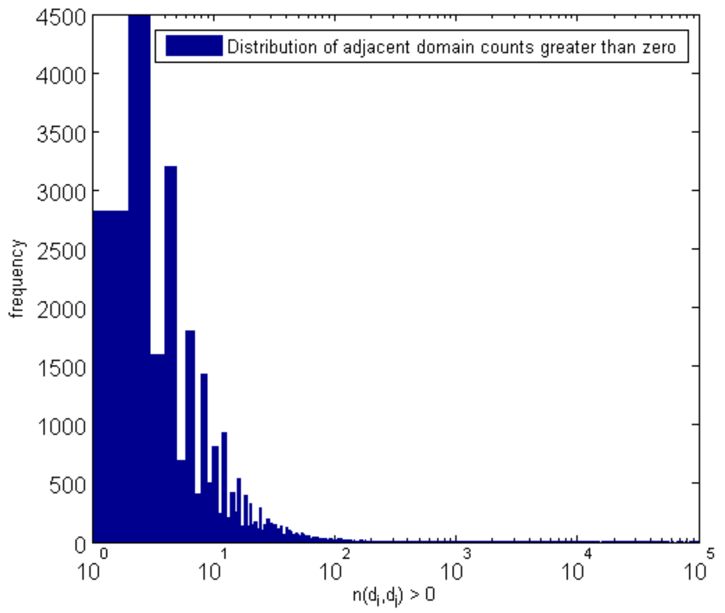


Figure 3-1 Support for the domain adjacency hypothesis. The figure represents the distribution of adjacent domain counts greater than zero. The x axis labels is log scaled and it represents all combinations of adjacent domain counts (how many times did domain $d_i$ appear to the left of domain $d_j$). The y axis represents the frequency.

Now we need to provide support for our hypothesis that the order of adjacent domains is important. In order to do that we count all domain combinations $d_i, d_j$ and their reverse $d_j, d_i$ to see how much they differ. As a measure of their difference we calculate

$\frac{abs(n(d_i,d_j)-n(d_j,d_i))}{n(d_i,d_j)+n(d_j,d_i)}$ for all domain combinations with $n(d_i,d_j)+n(d_j,d_i)>0$. Figure 3-2 shows that the vast majority of adjacent domain counts differ greatly from their flipped version. More than 75% of the cases are situated at the rightmost end of the histogram denoting where the difference between $n(d_i,d_j)$ and $n(d_j,d_i)$ is almost as large as the sum between them. Note that the counts at the x axis value of zero also contain the combinations $d_i,d_j$ with $i=j$ which still support our hypothesis. We conclude that there is significant support for our hypothesis that domain ordering in architectures is important and can provide valuable information for predicting protein domains. This finding is also in line with biological and chemical considerations as domains are usually asymmetrical in their amino acid composition. Our conclusion plays a key part in refuting the approach of using just co-occurrence as a measure of context which was prevalent in the literature.

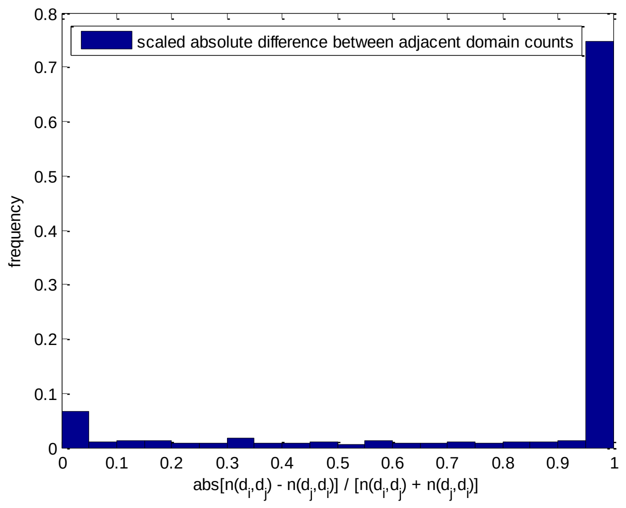


Figure 3-2 Support for the domain ordering hypothesis. The figure represents the distribution of the scaled absolute difference between counts of adjacent domains $d_i d_j$ and their reverse $d_j d_i$. The x axis represents the scaled difference of counts and the y axis the frequency.

Based on these findings, we focus our project around adjacency and ordering. In the final paragraphs of this section we give a high level description of our approach. The diagram in Figure 3-3 illustrates our approach based on an example. Let us assume that we have a sequence of amino acids and that SUPERFAMILY detected a list of domains through its HMM models. The output from SUPERFAMILY includes the domain names ($L$, $C$, $R$ and $U$) and the original E-values for each domain from the HMM alignment. For each domain in the architecture we find the closest domain in the left and in the right. For example domain $C$ has domain $L$ to the left and domain $R$ to the right. In order to make the E-value of domain $C$ more robust we look to see how often the combination $L$, $C$, $R$ appeared in other protein domain architectures. Depending on the occurrences in the protein domain architecture database the original HMM E-value will be multiplied by a factor making the prediction more or less significant.

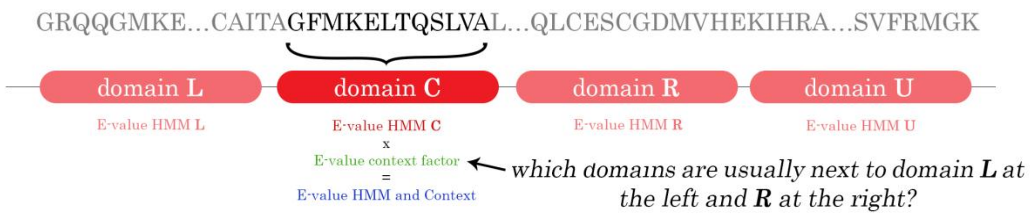


Figure 3-3: Diagram of a protein domain architecture. Source: author.

In the next subsection we formulate 6 scoring functions based on the adjacency and ordering principle.

## 3.2 Context Scoring Framework

We set ourselves three main requirements for our scoring framework in order for it to be successful:

1. It must be straightforward such that it is easily explained and accepted by researchers while capturing the most significant part of context information.
2. It must be statistically sound and easily incorporated with the current E-value calculations
3. It must be fast when implemented, being able to parse 60 million sequences without much extra computation time.

To fulfill the second requirement we start off from  the original score calculation method. The context scores currently calculated for HMM alignments take the form of sums of log odds ratios:

$$S(d) = \sum_i s(x_i, y_i)$$

where $s(a,b) = \log_2\left(\frac{p_{ab}}{q_a q_b}\right)$ can be taken from the substitution matrix.

Having also the first requirement in mind, the best way to formulate the new context framework would be an analytical function which can  be easily combined with the original scoring function. The second requirement would lead us to combine the new scoring functions by addition. Therefore we formulate the new score with context as follows:

$$S'(d) \stackrel{\text{def}}{=} S(d) + C(d_{left}, d) + C(d, d_{right})$$

where the contributions of each side of the context score is calculated as follows:

$$C(d_i, d_j) \stackrel{\text{def}}{=} \log_2\left(\frac{p(d_i, d_j)}{q(d_i)q(d_j)}\right)$$

where:

- $p(d_i, d_j)$ is the match probability of finding domain $d_i$ to the left of domain $d_j$ because they form togheter a structural/functional relation,
- $q(d_i)q(d_j)$ is the background probability of finding domain $d_i$ to the left of domain $d_j$ by chance, and
- $q(d_i)$ is the probability of finding domain $d_i$.

We also need to take into account cases when we do not have any domain appearing in the left, right or both. In that case the respective context score is set to zero and it does not bring any modifications to the original HMM score.

Now we need to define estimates of the match and background probabilities. We defined six variations of the scoring framework, and the differences between them come from the way the estimates are being defined.

The background probabilities are easier to define and they will be the same for all the scoring frunctions:

$$\hat{q}(d) = \frac{1}{n}$$

where $n$ is the number of domains found in nature, which is a number greater than 2000.

The match probability estimates are defined in three different variations.

The **first scoring framework** defines the match probability estimates as a ratio between the adjacency counts for each superfamily combination divided by the sum of adjacency counts for all superfamily combinations, with Laplace correction on top:

$$\hat{p}_1(d_i, d_j) = \frac{\hat{c}(d_i, d_j) + 1}{\sum_{ij} \hat{c}(d_i, d_j) + n^2}$$

where $n$ is the number of domains found in nature and $\hat{c}(d_i, d_j)$ are adjacency counts from existing proteins calculated as follows: $\hat{c}(d_i, d_j) = \sum_p \hat{n}_p(d_i, d_j)$, meaning that the counts are calculated as the number of times domain $d_i$ appeared at the left of domain $d_j$ in all proteins.

The additions of 1 to the numerator and $n^2$ to the denominator, also known as Laplace correction, help in avoiding the case where the probability estimate would be exactly zero which would lead to a log odds ration equal to minus infinity. In case the domain combination $d_i, d_j$ is not found in the existing protein domain architectures database then $\hat{c}(d_i, d_j) = 0$ and $\hat{p}_1(d_i, d_j) = \frac{1}{\sum_{ij} \hat{c}(d_i, d_j) + n^2}$.

It can be easily seen that the probabilities sum to one if we take all possible domain combinations: $\sum_{ij} \hat{p}_1(d_i, d_j) = 1$.

The **second scoring framework** defines the match probability estimates as follows:

$$\hat{p}_2(d_i, d_j) = \frac{\hat{c}(d_i, d_j) + \frac{1}{n^2}}{\sum_{ij} \hat{c}(d_i, d_j) + 1}$$

where $\hat{c}(d_i, d_j) = \sum_p \frac{\hat{n}_p(d_i, d_j)}{\hat{n}_p(d)}$, meaning that the counts are calculated as the number of times domain $d_i$ appeared at the left of domain $d_j$ in all proteins divided by the total number of domains existing in each protein. This takes into account that in longer proteins there is a higher chance of finding random combinations between domains. Therefore, for example we will consider that a domain adjacency occurrence conveys stronger context information if found in a two domain protein than if it occurs in a five domain protein.

At the same time the Laplace correction is different such that if combination $d_i, d_j$ is not found $\hat{p}_2(d_i, d_j) = \frac{1}{n^2(\sum_{ij} \hat{c}(d_i, d_j) + 1)}$ which would lead to a stronger negative context. In other words, the context score $C(d_i, d_j)$ would still be negative and greater in magnitude in the second framework than in the first framework when combination $d_i, d_j$ is not found: $C_2(d_i, d_j) < C_1(d_i, d_j)$.

The **third scoring framework** defines the match probability estimates as follows:

$$\hat{p}_3(d_i, d_j) = \frac{\hat{c}(d_i, d_j) + 1}{\sum_{ij} \hat{c}(d_i, d_j) + n^2}$$

where the adjacency counts $\hat{c}(d_i, d_j) = \sum_p \frac{\hat{n}_p(d_i, d_j)}{\hat{n}_p(d)}$.

Compared to the first and second frameworks, the third framework has the same adjacency counts as the second framework and the same Laplace correction as the first framework. This type of Laplace correction is such that if combination $d_i, d_j$ is not found $\hat{p}_3(d_i, d_j) = \frac{1}{\sum_{ij} \hat{c}(d_i, d_j) + n^2}$ which would lead to a weaker negative context.

Turning our attention back to the context scores $C$ it would be very useful to incorporate the information contained in the E-values of the left and right domains. The most intuitive way in which we can use that information would be if we give proportional weights to the context scores of the left $C(d_{left}, d)$ and right $C(d, d_{right})$ depending on the E-values $\mathrm{E}\left(S(d_{left})\right)$ and $\mathrm{E}\left(S(d_{right})\right)$. To put this into a mathematical formulation:

$$S''(d) \overset{\text{def}}{=} S(d) + w_{left} \cdot C(d_{left}, d) + w_{right} \cdot C(d, d_{right})$$

where

$$w_{left} = 2 \cdot \frac{\log_2 \mathrm{E}\left(S(d_{left})\right)}{\log_2 \mathrm{E}\left(S(d_{left})\right) + \log_2 \mathrm{E}\left(S(d_{right})\right)}, \text{ and}$$

$$w_{right} = 2 - w_{left}$$

In words, the weight for the left context score will be twice the ratio between the log of the E-value of the left domain and the sum of logs of the left and right E-values. The reason for using the log of E-value instead of just E-value is because the distribution of E-values has an exponential form and in most cases one of the weights $w$ will be almost zero and the other almost two. The logs will give more balanced weights.

Now the formulation for the context score $S'$ is a special case of $S''$ where the weights are both equal to one. The reason for multiplying the ratio in $S''$ with two is that the sum of the weights will be equal to 2 in both formulations.

For the sake of clear referencing we will call scoring frameworks using the weights as in $S''$ as follows:

- **fourth scoring framework** is the same as the first but using proportional weights,
- **fifth scoring framework** is the same as the second but using proportional weights, and
- **sixth scoring framework** is the same as the third but using proportional weights.

## 3.3 E-values

In this subsection we will show how we arrive at E-value context factors based on the context scores defined in the previous section.

The E-value of a match between an HMM model and a protein sequence can be calculated from the score value:

$$E\left(S''(d)\right) = Kmne^{-\lambda \cdot S''}$$

where $K$ is a parameter depending on the size of the database, $m$ is the length of the Hidden Markov Model, $n$ is the length of the amino acid sequence, $\lambda$ is a scale parameter equal to 1 in our case as we are using log odds ([10]), and $S''(d)$ is the modified score.

Our modified score is defined as:

$$S''(d) = S(d) + w_{left} \cdot C(d_{left}, d) + w_{right} \cdot C(d, d_{right})$$

Therefore we can separate the context part of the score as a factor to multiply with the original E-value:

$$E\left(S''(d)\right) = E\left(S(d)\right) \cdot e^{-\left(w_{left} \cdot C(d_{left}, d) + w_{right} \cdot C(d, d_{right})\right)}$$

By separating an E-value context factor we are avoiding the need to know the original score $S$. This is a great feature also because only the original E-values $E\big(S(d)\big)$ are stored in the SUPERFAMILY database but not the original scores $S$. Another desirable feature is that we avoid the need to know parameters $K$, $m$, and $n$.

## 3.4 **Data**

We used the SUPERFAMILY database for our project. Figure 3-4 shows the database schema that we will be using.

We have two important SQL statements which we will present in the next two subsection.

### 3.4.1 Generating protein domain architecture statistics

The SQL statement one generaters protein domain argitecture statistics. It returns all the possible protein domain architectures found in SUPERFAMILY in the following fomat:

- comb_index.comb which is a string specifying the codes of the domains. We will use this string in order to determine which domains are adjacent.
- comb_index.length, which is a number equal to the number of domains that constitute the protein domain architecture. We will use the length for scaling in some of our context functions
- len_supra.number, which is a number equal to the number of occurrences of that particular domain architecture in the database. We will use these counts in our scoring function

We only extract protein domain architectures with at least length two otherwise there is no context to be studied. The last two AND conditions are used for extracting protein domain architectures for a set of 2000 representative organisms.

SQL statement for  generating protein domain architecture statistics

```
SELECT comb_index.comb, comb_index.length, len_supra.number
        FROM comb_index,len_supra,genome
        WHERE comb_index.id=len_supra.supra_id
            AND comb_index.length >= 2
            AND len_supra.genome=genome.genome
            AND genome.include='y'";
```

We process the data obtained from the database with the scoring functions and obtain the hash table with the scores. The following table displays the output from the calculation procedure of the hash table for one of the scoring functions. Notice the first column contains codes for the superfamily on the left (heading sfL), the second column contains codes for the superfamily on the right (heading sfR), and the third column contains the actual score for the (sfL,sfR) combination. Notice that the first row of the table contains a record with sfL=0 and sfR=0. That row is reserved for the score for the case when a certain domain combination was never found before in the database (notice the extreme negative score of -23).

Display of the calculation of the hash table for one of the scoring functions

```
RUNNING saveContextScores:
 > Calculating number of domains observed:
   > n=2223
 > Calculating adjacency counts:
   > clr
   > c=8612980
 > Calculating context scores:
   > Clr
       sfL        sfR                     cLR
         0          0   -23.0380812536423427
    100879     100879    -0.2989962409829441
    100879      47576    -0.7541907902861432
    100879      52096    -0.0279021006579494
    100879      52540     1.1371027439061079
    100879      53271    -1.5791038657811118
    100879      53335    -0.8754970617254845
    100879      53383    -0.3160697518885972
    100879      55729    -0.1084743604274835
    100879      56019    -0.0079474966280937
    100879      56112    -1.5791038657811118
    100879      56672    -1.0645308430925755
…
```

### 3.4.2 Extracting HMM assignment data

The second SQL statement is used for extracting HMM assignment data in the following format:

- ass.sf, which is the superfamily code of the domain
- ass.evalue, which is the E-value of the domain assignment
- ass.region, which is a list of numbers which specify where in the protein was the domain detected

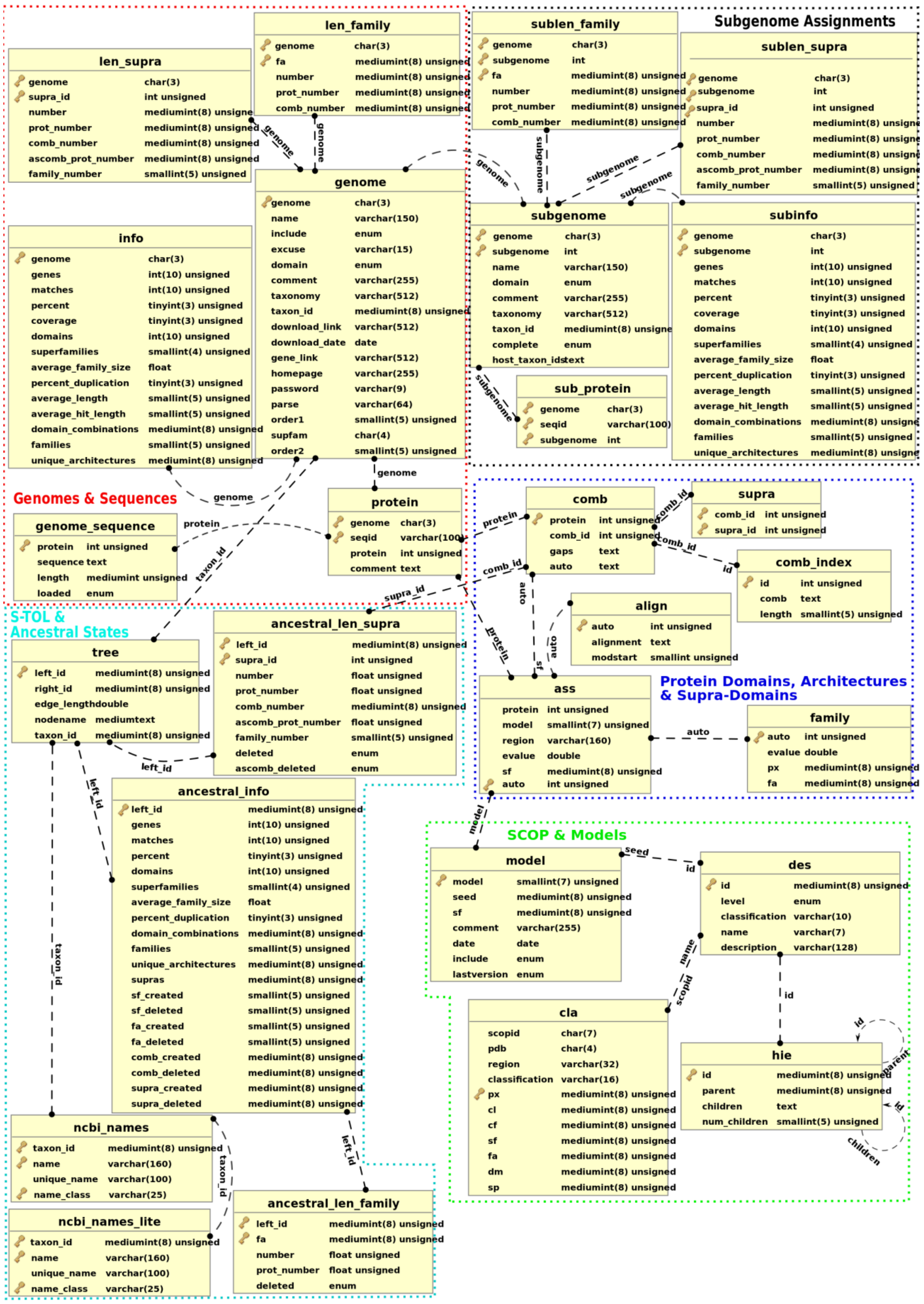
SQL statement for  extracting HMM assignment data

```
SELECT ass.sf,ass.evalue,ass.region
        FROM ass,protein,genome
        WHERE protein.genome=genome.genome
            AND genome.include='y'
            AND protein.seqid='$seqid'
            AND protein.protein=ass.protein;
```

We process the data extracted by the second SQL statement in order to be able to detect the order in which domains appear in an assignment. In the table below we displayed the result of processing the assignment data from SUPERFAMILY. Notice how each protein sequence is treated separately and the domains left (sfL), center (sfC), and right (sfR) are being placed in order. Each row in the table corresponds to a domain being analysed (sfC) and it's adjacent domains (sfL and sfR). Each row also has context scores attached to the specific domain combinations found. Notice how the context score (scoreL or scoreR) are equal to zero when we have no left or right context. The final two column represent the original E-value from the HMM models (eval1) and the modified E-value with our context score (eval2).

Display of the processing of assignment data from SUPERFAMILY

```
RUNNING assign(ENSPPYP00000000002):

     sfL       sfC       sfR    scoreL    scoreR     eval1     eval2
   57667     57667               17.23      0.00   8.6e-16   2.8e-17
             57667     57667      0.00     17.23   6.3e-15   2.0e-16
             57667                0.00      0.00   3.3e-11   3.3e-11

RUNNING assign(ENSPPYP00000000003):

     sfL       sfC       sfR    scoreL    scoreR     eval1     eval2
   57667     57667               17.23      0.00   7.3e-16   2.3e-17
             57667     57667      0.00     17.23   5.2e-15   1.7e-16
             57667                0.00      0.00   2.5e-11   2.5e-11

RUNNING assign(ENSPPYP00000000004):

     sfL       sfC       sfR    scoreL    scoreR     eval1     eval2
   57667     57667               17.23      0.00   2.7e-16   8.6e-18
             57667     57667      0.00     17.23   6.9e-15   2.2e-16
             57667                0.00      0.00   1.6e-08   1.6e-08
   57667     57667               17.23      0.00   5.1e-03   1.6e-04

RUNNING assign(ENSPPYP00000000005):

     sfL       sfC       sfR    scoreL    scoreR     eval1     eval2
             57667                0.00      0.00   2.8e-14   2.8e-14
             57667                0.00      0.00   4.6e-08   4.6e-08
   57667     57667               17.23      0.00   5.0e-04   1.6e-05

RUNNING assign(ENSPPYP00000000023):

     sfL       sfC       sfR    scoreL    scoreR     eval1     eval2
   57845     49899                9.53      0.00   5.9e-64   8.8e-65
             57850     57845      0.00     11.04   1.2e-20   1.3e-21
   57850     57845     49899     11.04      9.53   6.3e-15   1.0e-16

RUNNING assign(ENSPPYP00000000034):

     sfL       sfC       sfR    scoreL    scoreR     eval1     eval2
             47986     52540      0.00      8.66   2.7e-24   4.8e-25
   47986     52540                8.66      0.00   6.1e-10   1.1e-10
   47986    140856     52540    -23.41    -23.41   4.2e-02   4.9e+02
...
```



Genomes & Sequences
Subgenome Assignments
S-TOL & Ancestral States
Protein Domains, Architectures & Supra-Domains
SCOP & Models
len_supra
genome char(3)
supra_id int unsigned
number mediumint(8) unsigned
prot_number mediumint(8) unsigned
comb_number mediumint(8) unsigned
ascomb_prot_number mediumint(8) unsigned
family_number smallint(5) unsigned
len_family
genome char(3)
fa mediumint(8) unsigned
number mediumint(8) unsigned
prot_number mediumint(8) unsigned
comb_number mediumint(8) unsigned
sublen_family
genome char(3)
subgenome int
fa mediumint(8) unsigned
number mediumint(8) unsigned
prot_number mediumint(8) unsigned
comb_number mediumint(8) unsigned
sublen_supra
genome char(3)
subgenome int
supra_id int unsigned
number mediumint(8) unsigned
prot_number mediumint(8) unsigned
comb_number mediumint(8) unsigned
ascomb_prot_number mediumint(8) unsigned
family_number smallint(5) unsigned
info
genome char(3)
genes int(10) unsigned
matches int(10) unsigned
percent tinyint(3) unsigned
coverage tinyint(3) unsigned
domains int(10) unsigned
superfamilies smallint(4) unsigned
average_family_size float
percent_duplication tinyint(3) unsigned
average_length smallint(5) unsigned
average_hit_length smallint(5) unsigned
domain_combinations mediumint(8) unsigned
families smallint(5) unsigned
unique_architectures mediumint(8) unsigned
genome
genome char(3)
name varchar(150)
include enum
excuse varchar(15)
domain enum
comment varchar(255)
taxonomy varchar(512)
taxon_id mediumint(8) unsigned
download_link varchar(512)
download_date date
gene_link varchar(512)
homepage varchar(255)
password varchar(9)
parse varchar(64)
order1 smallint(5) unsigned
supfam char(4)
order2 smallint(5) unsigned
subgenome
genome char(3)
subgenome int
name varchar(150)
domain enum
comment varchar(255)
taxonomy varchar(512)
taxon_id mediumint(8) unsigned
complete enum
host_taxon_ids text
subinfo
genome char(3)
subgenome int
genes int(10) unsigned
matches int(10) unsigned
percent tinyint(3) unsigned
coverage tinyint(3) unsigned
domains int(10) unsigned
superfamilies smallint(4) unsigned
average_family_size float
percent_duplication tinyint(3) unsigned
average_length smallint(5) unsigned
average_hit_length smallint(5) unsigned
domain_combinations mediumint(8) unsigned
families smallint(5) unsigned
unique_architectures mediumint(8) unsigned
sub_protein
genome char(3)
seqid varchar(100)
subgenome int
genome_sequence
protein int unsigned
sequence text
length mediumint unsigned
loaded enum
protein
genome char(3)
seqid varchar(100)
protein int unsigned
comment text
comb
protein int unsigned
comb_id int unsigned
gaps text
auto text
supra
comb_id int unsigned
supra_id int unsigned
comb_index
id int unsigned
comb text
length smallint(5) unsigned
align
auto int unsigned
alignment text
modstart smallint unsigned
ass
protein int unsigned
model smallint(7) unsigned
region varchar(160)
evalue double
sf mediumint(8) unsigned
auto int unsigned
family
auto int unsigned
evalue double
px mediumint(8) unsigned
fa mediumint(8) unsigned
tree
left_id mediumint(8) unsigned
right_id mediumint(8) unsigned
edge_length double
nodename mediumtext
taxon_id mediumint(8) unsigned
ancestral_len_supra
left_id mediumint(8) unsigned
supra_id int unsigned
number float unsigned
prot_number float unsigned
comb_number mediumint(8) unsigned
ascomb_prot_number float unsigned
family_number smallint(5) unsigned
deleted enum
ascomb_deleted enum
ancestral_info
left_id mediumint(8) unsigned
genes int(10) unsigned
matches int(10) unsigned
percent tinyint(3) unsigned
domains int(10) unsigned
superfamilies smallint(4) unsigned
average_family_size float
percent_duplication tinyint(3) unsigned
domain_combinations mediumint(8) unsigned
families smallint(5) unsigned
unique_architectures mediumint(8) unsigned
supras mediumint(8) unsigned
sf_created smallint(5) unsigned
sf_deleted smallint(5) unsigned
fa_created smallint(5) unsigned
fa_deleted smallint(5) unsigned
comb_created mediumint(8) unsigned
comb_deleted mediumint(8) unsigned
supra_created mediumint(8) unsigned
supra_deleted mediumint(8) unsigned
ncbi_names
taxon_id mediumint(8) unsigned
name varchar(160)
unique_name varchar(100)
name_class varchar(25)
ncbi_names_lite
taxon_id mediumint(8) unsigned
name varchar(160)
unique_name varchar(100)
name_class varchar(25)
ancestral_len_family
left_id mediumint(8) unsigned
fa mediumint(8) unsigned
number float unsigned
prot_number float unsigned
deleted enum
model
model smallint(7) unsigned
seed mediumint(8) unsigned
sf mediumint(8) unsigned
comment varchar(255)
date date
include enum
lastversion enum
des
id mediumint(8) unsigned
level enum
classification varchar(10)
name varchar(7)
description varchar(128)
cla
scopid char(7)
pdb char(4)
region varchar(32)
classification varchar(16)
px mediumint(8) unsigned
cl mediumint(8) unsigned
cf mediumint(8) unsigned
sf mediumint(8) unsigned
fa mediumint(8) unsigned
dm mediumint(8) unsigned
sp mediumint(8) unsigned
hie
id mediumint(8) unsigned
parent mediumint(8) unsigned
children text
num_children smallint(5) unsigned
genome
subgenome
protein
taxon_id
comb_id
supra_id
id
auto
sf
model
seed
name
scopid
left_id
parent
children

Figure 3-4: SUPERFAMILY database schema. Source: http://supfam.org /SUPERFAMILY/pics/database_schema_1_75.png

## 3.5 **Benchmark**

This section explains how we conceived our benchmark and the difficulties we faced (subsection 3.5.1) and the way we planned the testing of our benchmark (subsection 3.5.2).

### 3.5.1 Concept

Normally when building a benchamark we need three elements: an original method, a modified method and a ground truth. Unfortunately there is no ground truth when dealing with protein sequences and superfamily annotations. Therefore we had to be creative when devising a benchmark.

We decided to use very strong and very weak assignments from the SUPERFAMILY database. Consiquently we consider ground truth correct assignments domains with an HMM E-value less than $10^{-8}$ and ground truth incorrect assignments those with an E-value greater than $10^{-2}$.

Our E-value context factor will act as a classifier of domains based only on the context information. A domain will be classified as positive if the E-value context factor is smaller than 1 and negative otherwise.

For clarity the following table explains how a confusion matrix for this classifier can be calculated:

| | E-value HMM: $E\big(S(d)\big)$ | E-value context factor: $e^{-\left(w_{left} \cdot C(d_{left}, d) + w_{right} \cdot C(d, d_{right})\right)}$ |
|---|---|---|
| True Positive if | $< 10^{-8}$ | $< 10^{0}$ |
| False Positive if | $> 10^{-2}$ | $< 10^{0}$ |
| True Negative if | $> 10^{-2}$ | $> 10^{0}$ |
| False Negative if | $< 10^{-8}$ | $> 10^{0}$ |

Table 3-1: Method for calculating the confusion matrix for the context classifier.
Source: author.

### 3.5.2 Testing

For testing if there is an improvement over the benchmark we took the following steps.

The first step is to calculate the satistics. We query the SUPERFAMILY database containing protein domain architectures for representative genomes (around two thousand organisms, or 60 million protein domain architectures) and UniProt. We iterate through each architecture and build up the adjacency counts $\hat{c}\big(d_i, d_j\big)$. Having the adjacency counts we can calculate $C\big(d_i, d_j\big)$ which we store in a hash table.

The second step is to calculate the E-value context factors for each protein in the database. To do this we iterate through the database taking each protein at a time and parse the

SUPERFAMILY assignment results containing: superfamily code, region of the protein where the superfamily was found, and the E-value of the assignment. For each superfamily annotated we search in the protein to see what is the superfamily to the left which is closest and the superfamily to the right which is closest. We consider a superfamily to act as context only if it has an E-value smaller than $10^{-4}$. Now we have a long list where each record contains: superfamily code left, E-value left, superfamily code center, E-value code center, superfamily code right and E-value code right. Having these records and the hash table with context scores $C(d_i, d_j)$ we can compute the E-value context factors for all of the six scoring frameworks.

Lastly, if we have the E-value context factors we can calculate preformance measures such as the ROC curve.

# 4 Results

In this chapter we discuss the results obtained. In the first section (4.1) we present the context score distributions, then we look at the E-value context factor distributions and afterwards we explain how the original E-values are combined with the context factors. In the second section (4.2) we calculate performance measures for the six scoring frameworks and we decide which one performs best. Here we also provide proof that adjacency and ordering information improves protein domain identification.

## 4.1 Score and E-value distributions

Before computing performance measures of the context scoring frameworks it is necessary to inspect how the context scores are distributed to see where the performance might come from and if the values are reasonable.

Figure 4-1 contains histograms for the context scores $C_1(d_i, d_j)$, $C_2(d_i, d_j)$ and $C_3(d_i, d_j)$ for all combinations if $d_i, d_j$ (i.e. $\forall i$ and $\forall j$). We start with the distribution for the first context scoring framework. The shape of the distribution is not surprisingly similar to the distribution of the adjacency counts, with a peak at the minimum and exponentially decreasing towards the maximum. We can see that most of the context scores will increase the E-values providing negative context and the maximum score is $\max_{\forall i,j} C_1(d_i, d_j) \cong 20$. There is a peak at the minimum of the distribution which is found for domain combinations $d_i, d_j$ not found in existing protein domain architectures ($\min_{\forall i,j} C_1(d_i, d_j) = -5.45$). We move on to analyse the second scoring framework. Here we see that the minimum of the distribution (for domain combinations that are not found in existing protein domain architectures) is much lower: $\min_{\forall i,j} C_2(d_i, d_j) = -23.03$. This is a very significant adjustment for the original E-value. The right tail of the second distribution is not too different from the first distribution with the maximum positive context being very similar $\max_{\forall i,j} C_2(d_i, d_j) \cong 19$. The third distribution is similar to the second but due to the different Laplace correction the minimum is much closer to the average of the distribution: $\min_{\forall i,j} C_2(d_i, d_j) = -1.45$. This leads to a more conservative context adjustment of the original E-value.

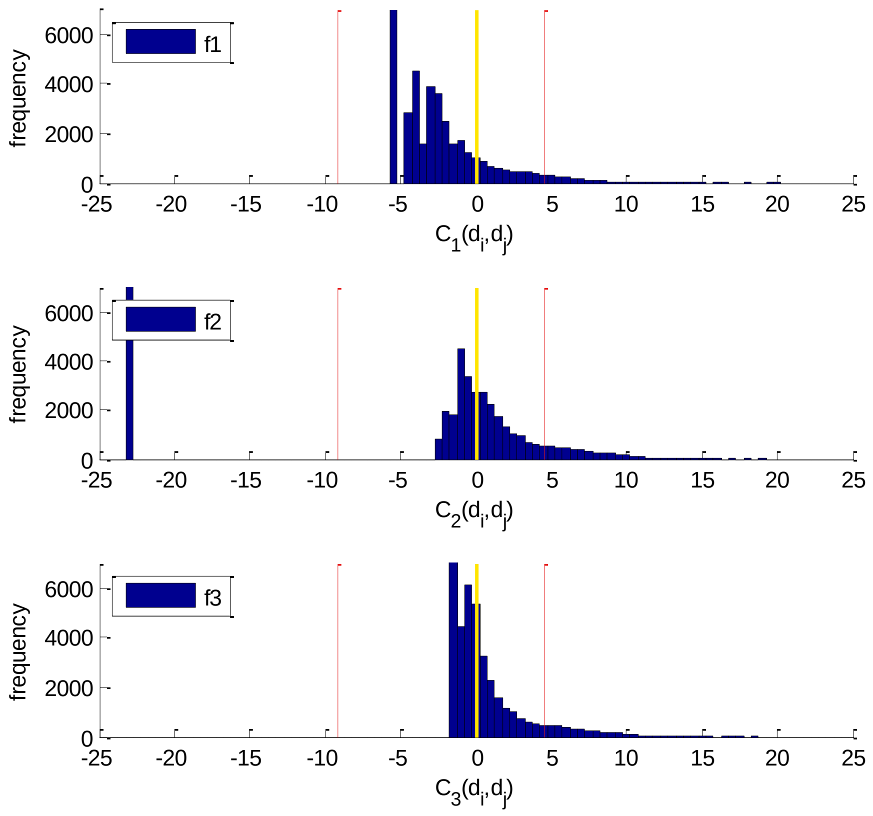


Figure 4-1 Distributions of the context scores $C(d_i, d_j)$ for scoring frameworks 1, 2 and 3 respectively. The $x$ axis represents the context scores and the $y$ axis the frequency. The values are obtained by taking all the possible combinations of $d_i, d_j$. The three vertical lines are drawn at threshold values $ln\left(\frac{E_{min}^{Th}}{E^{Th}}\right)$, 0 and $ln\left(\frac{E_{max}^{Th}}{E^{Th}}\right)$ respectively, where $E_{min}^{Th} = 10^{-8}$, $E^{Th} = 10^{-4}$ and $E_{max}^{Th} = 10^{-2}$. We use the threshold in the centre at $x = 0$ (plotted with orange) to distinguish between negative context (to the left of the zero threshold) which will demote a domain making it more insignificant and positive context which will promote a domain. The two sideward thresholds (plotted with red) mark the limit of having 'very' negative context score and 'very' positive context score. This means that if the context score passes one of the sideward thresholds then we are significantly altering the original HMM E-value. Source: histograms generated by the author in MATLAB

We now inspect the distributions of the E-value context factors from Figure 4-2. What is essentially different from Figure 4-1 is that the histogram is now based on the scores applied to the actual protein sequences. It is immediately visible that the distributions in the top row are very similar to the ones in the bottom row which leads us to conclude that adding the weights in the E-value context factors should not make a huge difference. Now taking each column in turn, we can see that frameworks 1 and 4 only provide negative context. Scoring frameworks 2 and 5 provide much more positive context (decreasing E-values due to context) than negative context (increasing E-values) which is not surprising because it is very unlikely to find domains in the database that rarely appeared in a combination with another domain. Also noteworthy is the fact that most of the values are outside the red vertical lines, meaning that the changes in E-value due to context will be important. Scoring frameworks 3 and 6 are very similar to frameworks 2 and 5 but due to a different Laplace correction the values are less extreme. These last scoring schemes will also generate an important change in the E-values.

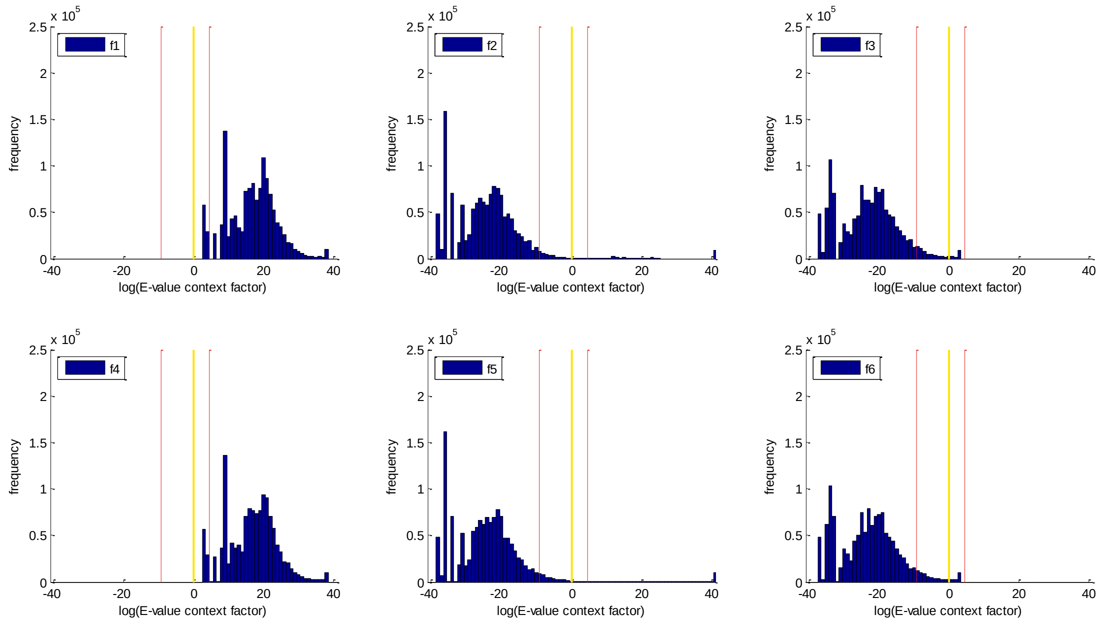


Figure 4-2 Distributions of E-value context factors. The three vertical lines are drawn at threshold values $ln\frac{E_{min}^{Th}}{E^{Th}}$, 0 and $ln\frac{E_{max}^{Th}}{E^{Th}}$ respectively, where $E_{min}^{Th} = 10^{-8}$, $E^{Th} = 10^{-4}$ and $E_{max}^{Th} = 10^{-2}$. We use the threshold in the centre at $x = 0$ (plotted with orange) to distinguish between negative context (to the left of the zero threshold) which will demote a domain making it more insignificant and positive context which will promote a domain. The two sideward thresholds (plotted with red) mark the limit of having 'very' negative context score and 'very' positive context score. Source: histograms generated by the author in MATLAB.

Now we exemplify how the original HMM E-values work together with the E-value context factors to generate the modified E-values. Figure 4-3 shows all these distributions together with vertical lines that mark three thresholds: $E_{min}^{Th}$, $E^{Th}$ and $E_{max}^{Th}$. The red distribution plots the original HMM E-value and the overlapping green distribution plots the E-value context factor distributions for the second scoring framework. By multiplying the values in the two distributions we arrive at the modified E-values which are plotted in the blue distribution. Ideally we would want to see values in the uncertainty region (between the lateral thresholds $E_{min}^{Th}$ and $E_{max}^{Th}$) move to the side. This is exactly what we see happening in the blue distribution.

In the following section we will see whether the direction of movement generated by the E-value context factors is the right one.

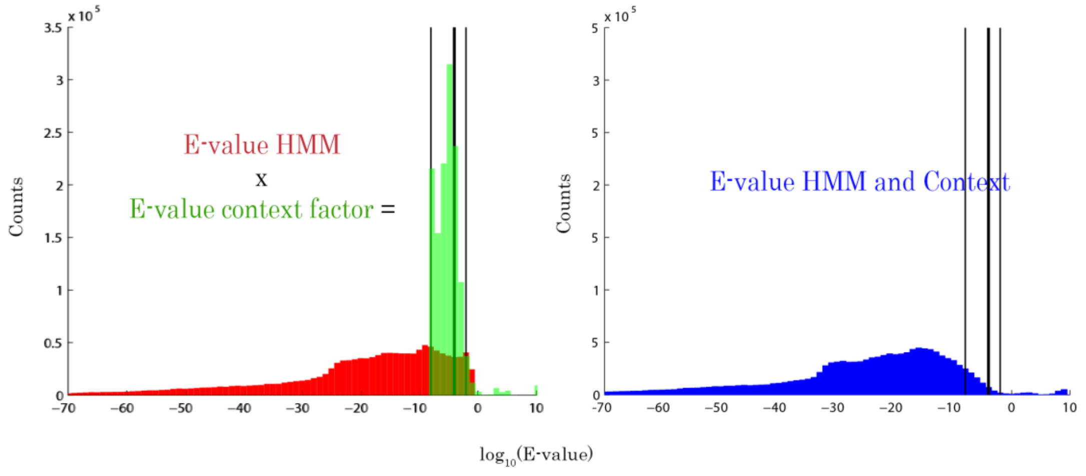


Figure 4-3: The distributions of the original E-values from HMM (red), Evalue context factors for the first scoring framework (green) and the new E-values with the context correction (blue). The $x$ axis is in $\log_{10}(Evalue)$ scale and the $y$ axis represents the frequency. Source: histograms generated in MATLAB by the author.

## 4.2 Choosing a context framework

In this section we first check if any of our context frameworks brings any improvement to current methods and we will choose the one that performs the best.

In order to achieve our goal we first plot the false positive rate against the true positive rate in a ROC plot. Figure 4-4 shows the ROC plots for all the six scoring frameworks. We are pleased to see that all functions perform significantly better than chance. The first observation we make is that frameworks 4-6 perform better than 1-3 which is not surprising. This supports the supposition that giving more weight to the more significant context increases the predictive power of our classifier. The second observation is that the first framework performs the poorest which confirms our supposition that adjacency counts need to be scaled by the number of domains in each protein. The last observation is that the fifth and sixth frameworks perform the best. There are two main arguments that would explain this increased performance. First, the adjacency counts are scaled with the domain counts in each protein. Second, more weight is given to whichever adjacent domain has a more significant E-value.

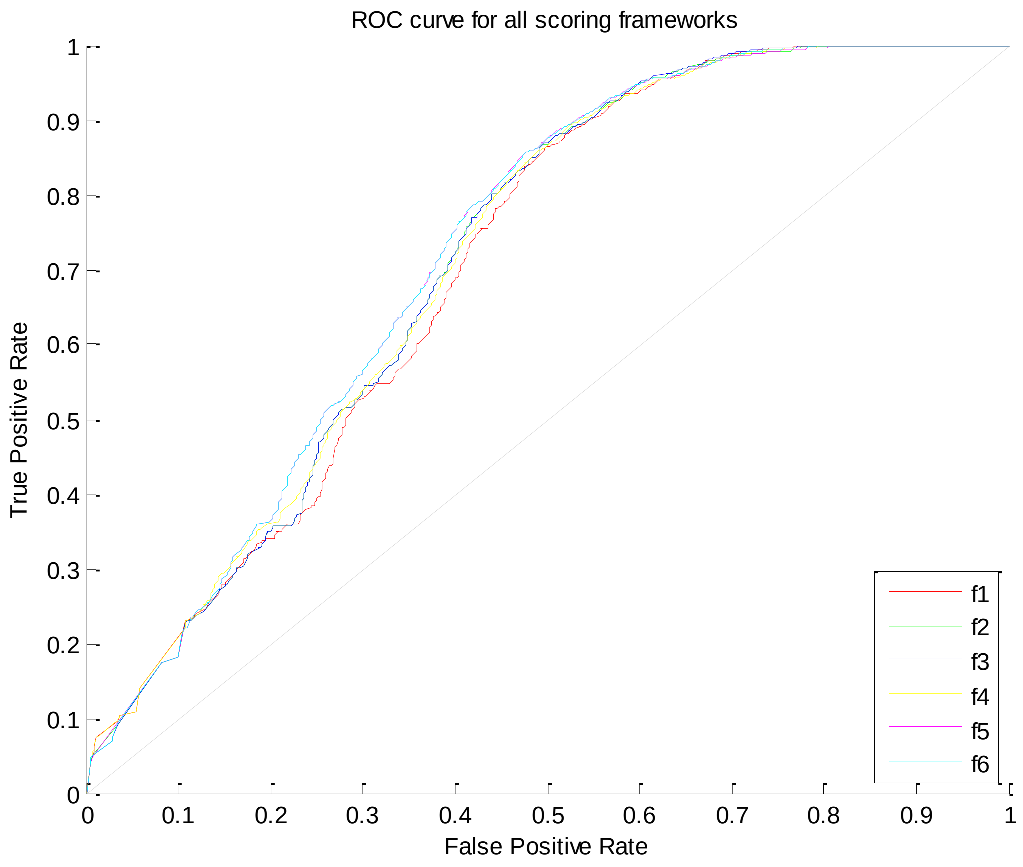


Figure 4-4: ROC curve for all the context scoring frameworks.

In order to properly rank the scoring frameworks we calculate the area under the ROC curve (Table 4-1).

| **Scoring Framework** | **Area under the ROC curve** | **Optimal Operating Point** | **TPR (FPR=0.01)** |
|---|---|---|---|
| f1 | 0.705 | [0.77, 0.99] | 0.0752 |
| f2 | 0.712 | [0.81, 0.99] | 0.0495 |
| f3 | 0.712 | [0.77, 0.99] | 0.0495 |
| f4 | 0.713 | [0.83, 0.99] | 0.0719 |
| f5 | **0.722** | [0.83, 0.99] | 0.0485 |
| f6 | **0.722** | [0.80, 0.99] | 0.0494 |

Table 4-1 Area under the ROC curve, optimal operating point and True Positive Rate at a 1% False Positive Rate for all the scoring frameworks

It is therefore evident that all of the scoring functions would be able to improve protein domain identification and they can be successfully added to the SUPERFAMILY analysis pipeline. Nevertheless the methods with which use different weights for left and right context perform better and also the ones that use scaled counts.

## 4.3 **Discussion**

Our project proves that context information is useful based on testing on much more representative data than any of the previous papers on the subject. The method is tested on 2000 representative organisms or 60 million protein sequences while the previous papers use maximum 8 organisms.

This is also the first paper so far that uses domain adjacency and ordering information rather than just co-occurrence. This approach is justified by observing that adjacent domains have extremely different counts if we change the order of the domain.

At the same time it is the first paper that uses superfamily level annotations rather than family level annotations and adds futher support to the finding that context information is useful for protein domain annotation. Also, there are fewer superfamilies than families in nature therefore the conclusion that context information is a valuable addition is more robust.

Another finding of this thesis is that it does not make a large difference if we calculate the statistics on UniProt or on the representative organisms (it doesn’t matter if we use just representative species or the whole genomes).

# 5 Evaluation of the project

Our project had the aim of improve protein domain identification using context information, more specifically domain adjacency and ordering.

We will take each objective one by one and discuss how much we achieved.

The first objective was to generate statistics of protein domain architectures from SUPERFAMILY. We can say that we successfully achieved this objective. First we demonstrated that protein domains appear in nature in only a few combinations next to each other. Out of all the possible domain combinations, less than 1% appeared at least once in the SUPERFAMILY databases. This gave strong support to our hypothesis that domain adjacency could be used as strong context information. Second, we provided support for our hypothesis that the order of adjacent domains is important. In order to do that we counted all domain combinations $d_i, d_j$ and their reverse $d_j, d_i$ to see how much they differed. We found that the vast majority of adjacent domain counts differed greatly from their flipped version. Third, we used the two findings to calculate the exact values which were later used in our scoring functions.

The second objective was to formulate several scoring functions. We achieved this goal by formulating six types of scoring functions which were calculated as log odds ratios, being very easy to integrate in the classical sum of log odds scoring framework We went one step further and we simplified the calculation by transforming them into E-value factors. The first scoring function used adjacency counts from the SUPERFAMILY databases, the second used scaled adjacency counts with the total number of domains in each protein and the third used a different Laplace correction than the second. The scoring functions four to six were similar with one to three but they give a higher weight to whichever adjacent domain is more significant.

The third objective was to build a benchmark to estimate if there is any improvement over SUPERFAMILY. Due to the lack of a ground truth we used extremely significant and insignificant assignments from SUPERFAMILY and labelled them as true hits / false hits. Using this benchmark we proved that context information was giving similar predictions to the ones given by comparing the amino acids inside the domains. All the scoring functions could be used to improve the current results that the SUPERFAMILY analysis pipeline outputs.

The only objective which we did not achieve as it was unrealistic for the time frame that we had was to compare our method directly with the most successful competing method in the literature, dPUC [21] which uses domain co-occurrence rather than adjacency and it uses the family level classification from Pfam rather than superfamily classification. To do this we would have had to adapt the dPUC framework and then compare it with ours on our dataset of SUPERFAMILY domains. That would indeed have been quite difficult as dPUC uses Pfam annotations which are at the family level rather than at the superfamily level. Nevertheless the reason for not taking such a convoluted approach was the fact that we showed that domain ordering plays a very important role, thus contradicting the co-occurrence approach. As the project progressed we did not see a big gain in investing time in achieving this objective which we set out when writing the literature review.

In terms of deliverables we did produce an algorithm which enhances prediction of domains using context-dependent priors, optimised and validated on real data. We also built a valid benchmark based on the data available in the SUPERFAMILY databases.

We developed the framework with integration in the existing code from SUPERFAMILY in mind. That is why the implementation part is done in Perl. A basic integration would result in less than 300 lines of code comprised from a subroutine that calculates the scores and stores them in a serialized hash table and another subroutine that reads the hash table and applies the E-value context factors to the existing E-values from SUPERFAMILY. Our code is able to analyse 60 million sequences in a reasonable time frame. The extra time it takes to analyse a single protein sequence is negligible. We chose MATLAB for calculating the statistics and for comparing the performance of the different scoring schemens which we think was the right decision, reducing the potential coding time spent on Perl by half.

# 6 Further Research

There are plenty of directions for further improving the method presented in this thesis and the assessment of its contribution. In this chapter we will take each potential improvement and explain it in detail.

1. The first direction is related to comparing our method with the most successful competing method in the literature, dPUC [21]. To do this we would have had to adapt the dPUC framework and then compare it with ours on our dataset of SUPERFAMILY domains. That would indeed have been quite difficult as dPUC uses Pfam annotations which are at the family level rather than at the superfamily level. Nevertheless the reason for not taking such a convoluted approach was the fact that we showed that domain ordering plays a very important role, thus refuting the co-occurrence approach.

2. The second improvement would be to use ends of proteins, also called N-terminus and C-terminus, and treat them as separate domains which can provide further context information. Therefore a domain that always apears by itself in proteins would have context scoring that rewards the situations when it is found alone in a protein sequence.

3. The third improvement is related to the ground truth when testing which method performs better: SUPERFAMILY of SUPERFAMILY with context. If we shuffle a protein many times we would have a high degree of confidence that the shuffled protein does not belong to any supperfamily. Having this information we can then calculate the false positives and the true negatives. The difficulty in this approach would be the computational time, first due to the process of shuffling and then running all the HMM models on the shuffled proteins. We considered this improvement to be beyond the scope of our project as this would not be feasible computationally if we wanted to test the method on a large number of organisms.

4. The fourth improvement regards overlapping domains. When we feed in a sequence to the SUPERFAMILY analysis pipeline multiple models each related to a different superfamily may match a particular region in the protein sequence, each match having a different E-value. When finding overlapping assignments SUPERFAMILY returns only the most significant result (with the smalles E-value). If we would modify the analysis pipeline such that it returns all the matches including overlapping domains then we can apply the context factor to all the matches and may be able to promote a domain which would not be the most significant hit otherwise. We did not take this avenue due to exponentially increasing data storage and parsing requrements. Luckily the context factors would be fast enough to compute even for this challenging task.

5. The fifth issue which would benefit from further research is the detection of adjacent domains in a query amin oacid sequence. Currently a domain is considered adjacent to another domain without regard to the gap length and if there is any overlap, the adjacency is considered invalid. One approach would be to put a cap on the length of gap between the gap for the adjacency to be considered valid. Another issue is dealing

with multiple partial matches between a HMM model and a an aminoacid sequence. Currently we take the first partial match but there are other ways of doing this.

6. The sixth point which is worth investigating is the effect of limitting the allowed gap length in the protein domain architectures used for building the context scores table.

7. Currently we are only using the closest adjacent domains to the left and to the right of each domain which we are investigating. The seventh idea worth considering is to include domains further appart in the context information. It is unlikely that domains further than the second would bring any added value. At the same time it seems a good idea to give less weight to context information farther apart.

8. The seventh improvement regards the proportion of insignificant domains present in the assignment database. This task can be achieved by extending the existing SUPERFAMILY database with assignments with E-values greater than 0.1. Consequently we would have more domains which we can consider as ground truth negative.

9. As a ninth improvement we cound do extra testing to make sure that we always build the statistics on a different dataset than the one we are testing on (for example 10-fold cross validation). The reason for not doing this in this thesis is that there is proof that protein domain architectures are smilar across organisms so there is not too much potential for finding greatly varying results from what we found.

10. Lastly, regarding implementation, a further addition to the project would be full deployment on the SUPERFAMILY servers. This would be easily achived from a programming perspective as the code of this thesis was developped in Perl with integration in mind. However it would be a challenging software testing task which is beyond the scope of this thesis.

# 7 Conclusions

Our project proves that context information is useful based on testing on much more representative data than any of the previous papers on the subject. The method is tested on 2414 genomes of representative organisms or more than 60 million protein sequences while the previous papers used maximum 8 organisms.

This is also the first paper so far that uses domain adjacency and ordering information rather than just co-occurrence. This approach is justified by observing that adjacent domains have extremely different counts if we change the order of the domain.

At the same time it is the first paper that uses superfamily level annotations rather than family level annotations and adds futher support to the finding that context information is useful for protein domain annotation. Also, there are fewer superfamilies than families in nature therefore the conclusion that context information is a valuable addition is more robust.

We have also built our own benchmark for testing the performance of our scoring functions given the available information contained within the SUPERFAMILY databases.

The result of this project was the proof that we were able to improve a state of the art domain detection framework. By doing that we are not just showing a proof of principle but we have added real value to the scientific field.

# 8 Bibliography

[1] C. Vogel, C. Berzuini, M. Bashton, J. Gough, and S. a Teichmann, “Supra-domains: evolutionary units larger than single protein domains.,” *Journal of molecular biology*, vol. 336, no. 3, pp. 809-23, Feb. 2004.

[2] D. Wilson et al., “SUPERFAMILY--sophisticated comparative genomics, data mining, visualization and phylogeny.,” *Nucleic acids research*, vol. 37, no. Database issue, pp. D380-6, Jan. 2009.

[3] A. Bateman et al., “The Pfam protein families database.,” *Nucleic acids research*, vol. 32, no. Database issue, pp. D138-41, Jan. 2004.

[4] I. Letunic, T. Doerks, and P. Bork, “SMART 6: recent updates and new developments.,” *Nucleic acids research*, vol. 37, no. Database issue, pp. D229-32, Jan. 2009.

[5] P. D. Thomas, “PANTHER: a browsable database of gene products organized by biological function, using curated protein family and subfamily classification,” *Nucleic Acids Research*, vol. 31, no. 1, pp. 334-341, Jan. 2003.

[6] E. Koonin, “Sequence-evolution-function: computational approaches in comparative genomics,” 2003.

[7] M. Y. Guermeur, “Thèse de doctorat Discipline : Informatique Juliana Silva Bernardes l ’ annotation fonctionnelle et la classification des,” 2012.

[8] M. Bashton and C. Chothia, “The geometry of domain combination in proteins.,” *Journal of molecular biology*, vol. 315, no. 4, pp. 927-39, Jan. 2002.

[9] G. Apic, J. Gough, and S. a Teichmann, “Domain combinations in archaeal, eubacterial and eukaryotic proteomes.,” *Journal of molecular biology*, vol. 310, no. 2, pp. 311-25, Jul. 2001.

[10] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998, pp. 36-45.

[11] S. Astchul, W. Gish, W. Miller, and E. Myers, “Basic local alignment tool,” *J. Mol. Biol*, 1990.

[12] W. R. Pearson, “Rapid and sensitive sequence comparison with fastp and fasta,” *Methods in enzymology*, vol. 183. pp. 63-98, Jan-1990.

[13] S. Karlin and S. F. Altschul, “Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes.,” *Proceedings of the*

*National Academy of Sciences of the United States of America*, vol. 87, no. 6, pp. 2264-8, Mar. 1990.

[14] N. Bergman, “Comparative Genomics: Volumes 1 and 2,” 2007.

[15] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, 1989.

[16] C. Burges, “Advances in kernel methods: support vector learning,” 1999.

[17] S. Muggleton, “Inductive logic programming: Theory and methods,” *The Journal of Logic Programming*, 1994.

[18] L. Coin, A. Bateman, and R. Durbin, “Enhanced protein domain discovery by using language modeling techniques from speech recognition.,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 8, pp. 4516-20, Apr. 2003.

[19] F. Beaussart, J. Weiner, and E. Bornberg-Bauer, “Automated Improvement of Domain ANnotations using context analysis of domain arrangements (AIDAN).,” *Bioinformatics (Oxford, England)*, vol. 23, no. 14, pp. 1834-6, Jul. 2007.

[20] N. Terrapon, O. Gascuel, E. Maréchal, and L. Bréehélin, “Detection of new protein domains using co-occurrence: application to Plasmodium falciparum.,” *Bioinformatics (Oxford, England)*, vol. 25, no. 23, pp. 3077-83, Dec. 2009.

[21] A. Ochoa, M. Llinás, and M. Singh, “Using context to improve protein domain identification.,” *BMC bioinformatics*, vol. 12, no. 1, p. 90, Jan. 2011.

# 9 Appendix

## Main Subroutines for connecting with the MySQL Database (Perl)

```
#!/usr/bin/perl -w
use strict;
use DBI;
use DBD::mysql;
use Data::Dumper;
use Storable;
use List::Util qw[min max];
my $lambda = 0.2;
my $evalTh_Min = 10**(-6);
my $evalTh = 10**(-4);
my $evalTh_Max = 10**(-2);
my $fnContextScores = "ContextScores.data";
my $fnOut = "main.out";
my $start_run = time();
print "\n======================= main.pl SQL =======================\n";
# QUERY
my $sqlConnect = DBI->connect("dbi:mysql:superfamily:supfam2.cs.bris.ac.uk",
"rb1134", "1mai1986");
# CALCULATE & SAVE SCORES
# saveContextScores($sqlConnect, $fnContextScores);
# ASSIGN
assignAll($sqlConnect, $fnContextScores, $fnOut, $lambda, $evalTh_Min, $evalTh,
$evalTh_Max);
my $run_time = time() - $start_run;
print "\n======================= $run_time seconds =======================\n\n";
# SUB ---------------------------------------------------------------------
sub assignAll {

    # VARIABLES
    my $sqlConnect = shift;
    my $fnContextScores = shift;
    my $fnOut = shift;
    my $lambda = shift;
    my $evalTh_Min = shift;
    my $evalTh = shift;
    my $evalTh_Max = shift;
    my $seqid;
    my $TP = 0; my $FP = 0; my $TN = 0; my $FN = 0;
    my $dS_TP = 0; my $dS_FP = 0; my $dS_TN = 0; my $dS_FN = 0;
    my $aux3;
    # RETRIEVE SCORES
    my %Clr = %{retrieve $fnContextScores};
    # OUTPUT FILE
    open(my $fhOut, ">", $fnOut) or die "cannot open > $fnOut: $!";
    printf $fhOut "%8s%8s%8s%10s%10s%10s\n",
        "sfL", "sfC", "sfR", "evalL", "evalC", "evalR";
    # ASSIGN
    #$seqid = 'ENSPPYP00000002560'; $seqid = 'vb|PHUM607640-
PA|EEB20240.1|hypothetical';
    #assign($sqlConnect, \%Clr, $seqid, $evalTh, \$TP, \$FP, \$TN, \$FN,
\$dS_TP, \$dS_FP, \$dS_TN, \$dS_FN, $lambda);
    my $sqlSeqid0 = "SELECT protein.seqid, COUNT(ass.sf)
        FROM ass,protein
        WHERE protein.genome='of'
            AND protein.protein=ass.protein
        GROUP BY protein.seqid
        HAVING COUNT(ass.sf)>=2";
    my $sqlSeqid = "SELECT protein.seqid,COUNT(ass.sf)
```

```
        FROM ass,protein,genome
        WHERE protein.genome=genome.genome
            AND genome.include='y'
            AND protein.protein=ass.protein
        GROUP BY protein.seqid
        HAVING COUNT(ass.sf)>=2";
    my $sqlSeqidH = $sqlConnect->prepare($sqlSeqid);
    $sqlSeqidH->execute();
    $sqlSeqidH->bind_columns(\$seqid,\$aux3);
    while($sqlSeqidH->fetch()) {
        assign($sqlConnect, \%Clr, $fhOut, $seqid, $evalTh_Min, $evalTh,
$evalTh_Max, \$TP, \$FP, \$TN, \$FN,
            \$dS_TP, \$dS_FP, \$dS_TN, \$dS_FN, $lambda);
    }
    my $dS_pen = 1;
    my $dS_improvement = ($dS_TP - $dS_pen*$dS_FP - $dS_TN + $dS_pen*$dS_FN) /
max($TP + $FP + $TN + $FN, 1);
    printf "\n%6s %9s %7s %8s %9s %7s %8s %9s %7s %8s %9s %7s %8s %7s %8s\n",
        "lambda", "TP", "S+", "E*", "FP", "S+", "E*", "TN", "S+", "E*", "FN",
"S+",
        "E*", "S+", "E*";
    printf "%6.2f %9d %7.1f %8.1e %9d %7.1f %8.1e %9d %7.1f %8.1e %9d %7.1f
%8.1e %7.1f %8.1e\n",
        $lambda,
        $TP, $dS_TP/max($TP,1), exp(-$lambda*$dS_TP/max($TP,1)),
        $FP, $dS_FP/max($FP,1), exp(-$lambda*$dS_FP/max($FP,1)),
        $TN, $dS_TN/max($TN,1), exp(-$lambda*$dS_TN/max($TN,1)),
        $FN, $dS_FN/max($FN,1), exp(-$lambda*$dS_FN/max($FN,1)),
        $dS_improvement, exp(-$lambda*$dS_improvement);
    close $fhOut;
}
# -------------------------------------------------------------------------

# SUB ---------------------------------------------------------------------
sub assign {
    # VARIABLES
    my $sqlConnect = shift;
    my $Clr = shift;
    my $fhOut = shift;
    my $seqid = shift;
    my $evalTh_Min = shift;
    my $evalTh = shift;
    my $evalTh_Max = shift;
    my $tp = shift; my $fp = shift; my $tn = shift; my $fn = shift;
    my $dS_TP = shift; my $dS_FP = shift; my $dS_TN = shift; my $dS_FN = shift;
    my $lambda = shift;
    my ($hSf, $hEval, $hReg);
    my (@sfL, @sfC, @sfR, @evalL, @evalC, @evalR, @evalC2, @regL, @regR,
@scoreL, @scoreR);
    my (@aux1, @aux2);
    my ($r, $i, $i1, $i2);
    my ($regR_max, $regL_min);
    printf "RUNNING assign($seqid):\n";
    # QUERY ASSIGN TABLE
    my $sqlAss0 = "SELECT ass.sf,ass.evalue,ass.region
        FROM ass,protein
        WHERE protein.genome='of'
            AND protein.seqid='$seqid'
            AND protein.protein=ass.protein";
    my $sqlAss = "SELECT ass.sf,ass.evalue,ass.region
        FROM ass,protein,genome
        WHERE protein.genome=genome.genome
            AND genome.include='y'
            AND protein.seqid='$seqid'
            AND protein.protein=ass.protein";
    my $sqlAssH = $sqlConnect->prepare($sqlAss);
    $sqlAssH->execute();
    $sqlAssH->bind_columns(\$hSf, \$hEval, \$hReg);
    # LOOP THROUGH RESULTS
    while($sqlAssH->fetch()) {
        @aux1 = split(',', $hReg);
        @aux2 = split(/-/, $aux1[0]);
        push (@sfC, $hSf);
        push (@evalC, $hEval);
        push (@regL, $aux2[0]);
        push (@regR, $aux2[1]);
        push (@sfL, "");
        push (@evalL, "");
        push (@sfR, "");
        push (@evalR, "");
        push (@scoreL, 0);
        push (@scoreR, 0);
        push (@evalC2, 0);
    }
    # FIND LEFT & RIGHT DOMAINS
    $r = @sfC;
    for ($i1 = 0; $i1 < $r; $i1++) {
        $regR_max = -1000000;
        $regL_min = 1000000;
        for ($i2 = 0; $i2 < $r; $i2++) {
            if ($evalC[$i2] < $evalTh) {
                if (($regL[$i1] > $regR[$i2]) && ($regR[$i2] > $regR_max)) {
                    $regR_max = $regR[$i2];
                    $sfL[$i1] = $sfC[$i2];
                    $evalL[$i1] = $evalC[$i2];
                }
                if (($regR[$i1] < $regL[$i2]) && ($regL[$i2] < $regL_min)) {
                    $regL_min = $regL[$i2];
                    $sfR[$i1] = $sfC[$i2];
                    $evalR[$i1] = $evalC[$i2];
                }
            }
        }
    }
    # ADD CONTEXT SCORES
    for ($i = 0; $i < $r; $i++) {
        # Left
        if ($sfL[$i] ne "") {
            $scoreL[$i] = $Clr->{$sfL[$i]}{$sfC[$i]};
            if (!defined $scoreL[$i]) {$scoreL[$i] = $Clr->{0}{0};}
        }

        # Right
        if ($sfR[$i] ne "") {
            $scoreR[$i] = $Clr->{$sfC[$i]}{$sfR[$i]};
            if (!defined $scoreR[$i]) {$scoreR[$i] = $Clr->{0}{0};}
        }
        # Modified E-value
        $evalC2[$i] = $evalC[$i] * exp( -$lambda * ($scoreL[$i] + $scoreR[$i])
);

        if (($sfL[$i] ne "") && ($sfR[$i] ne "")) {

            # Performance Measures
            if ($evalC[$i]<$evalTh_Min && $evalC2[$i]<$evalTh_Min) {
                $$tp++;
                $$dS_TP += $scoreL[$i]+$scoreR[$i];
            } elsif ($evalC[$i]<$evalTh_Min && $evalC2[$i]>=$evalTh_Max) {
                $$fn++;
                $$dS_FN += $scoreL[$i]+$scoreR[$i];
            } elsif ($evalC[$i]>=$evalTh_Max && $evalC2[$i]<$evalTh_Min) {
                $$fp++;
                $$dS_FP += $scoreL[$i]+$scoreR[$i];
            } elsif ($evalC[$i]>=$evalTh_Max && $evalC2[$i]>=$evalTh_Max) {
                $$tn++;
                $$dS_TN += $scoreL[$i]+$scoreR[$i];
            };
            # Print
            printf $fhOut "%8s%8s%8s%10.2e%10.2e%10.2e\n",
                $sfL[$i], $sfC[$i], $sfR[$i],
                $evalL[$i], $evalC[$i], $evalR[$i];
        }
    }

}
# ------------------------------------------------------------------------

# SUB --------------------------------------------------------------------
sub saveContextScores {
    my $connect = shift;
    my $fileName = shift;
    my $comb;
    my $length;
    my $number;
    my @combArray;
    my @combArrayNoGaps;
    my $i;
    my $c = 0;
    my $clr;
    my $Clr;
    printf "RUNNING saveContextScores:\n";
    # n
    printf " > Calculating number of domains observed:\n";
    my $hSf;
    my $n = 0;
    my $sqlN_h = $connect->prepare("SELECT DISTINCT sf FROM cla");
    $sqlN_h->execute();
    $sqlN_h->bind_columns(\$hSf);
    while($sqlN_h->fetch()) {
        $n++;
    }
    printf "   > n=%d\n", $n;
    # clr
    printf " > Calculating adjacency counts:\n";
    printf "   > clr\n";
    my $sql0 =
        "SELECT comb, length, number
            FROM comb_index,len_supra
            WHERE comb_index.id=len_supra.supra_id
                AND comb_index.length >= 2
                AND genome='up'";
    my $sql =
        "SELECT comb_index.comb, comb_index.length, len_supra.number
            FROM comb_index,len_supra,genome
            WHERE comb_index.id=len_supra.supra_id
                AND comb_index.length >= 2
                AND len_supra.genome=genome.genome
                AND genome.include='y'";
# select comb, length, sum(number) from comb_index,len_supra,genome where
comb_index.id=len_supra.supra_id and comb='56719,101904' and
len_supra.genome=genome.genome AND genome.include='y';
    my $sqlH = $connect->prepare($sql);
    $sqlH->execute();
    $sqlH->bind_columns(\$comb, \$length, \$number);
    while($sqlH->fetch()) {
        @combArray = split(/,/, $comb);
        @combArrayNoGaps = grep {$_ ne '_gap_'} @combArray;
        if (@combArrayNoGaps>=2) {
            for ($i=0; $i<@combArrayNoGaps-1; $i++) {
                $clr->{$combArrayNoGaps[$i]}{$combArrayNoGaps[$i+1]} +=
$number/$length;
```

```
                $c += $number/$length
            }
        }
    }
    printf "   > c=%d\n", $c;
    # Clr
    printf " > Calculating context scores:\n";
    printf "   > Clr\n";
    foreach my $k1 (sort keys %$clr) {
        foreach my $k2 (sort keys %{$clr->{$k1}}) {
            $Clr->{$k1}{$k2} = log(($clr->{$k1}{$k2}*$n*$n + 1) / ($c+1)) /
log(2);
        }
    }
    $Clr->{0}{0} = log(1 / ($c+1)) / log(2);
    store $Clr, $fileName;
    ClrPrint($Clr);
    # return $Clr;
}
# --------------------------------------------------------------------------

# SUB ----------------------------------------------------------------------
sub ClrPrint {
    my $clr = shift;
    printf "%10s %10s %22s\n", "sfL", "sfR", "Plr";
    foreach my $key1 (sort keys %$clr) {
        foreach my $key2 (sort keys %{$clr->{$key1}}) {
            printf "%10s %10s %22.16f\n", $key1, $key2, $clr->{$key1}{$key2};
        }
    }
    printf "\n";
}
# --------------------------------------------------------------------------

# SUB ----------------------------------------------------------------------
sub mRowsCols {  # return number of rows and columns
    my ($r_mat)  = @_;
    my $num_rows = @$r_mat;
    my $num_cols = @{$r_mat->[0]}; # Assume all rows have an equal no. of
columns.
    ($num_rows, $num_cols);
}
# --------------------------------------------------------------------------
```