SISSA

Scuola
Internazionale
Superiore di
Studi Avanzati

Physics Area – PhD course in

Physics and Chemistry of Biological Systems

Unsupervised protein family classification by Density Peak clustering

Candidate:

Elena Tea Russo

Advisor:

Alessandro Laio

(SISSA)

Co-advisor:

Marco Punta

(IRCCS Ospedale San Raffaele)

Academic Year 2019-20

Faber est
suae
quisque
fortunae(?)

Contents

List of Abbreviations

MC - Metacluster.

GTA (of a MC protein region) - Ground Truth Architecture. *The set of Pfam families overlapping with a MC protein region.*

DA (of a MC) - Dominant Architecture. *The most common GTA of a MC. This can be defined both ad a family or at a clan level.*

%DAF (of a MC) - % Dominant Architecture (Family). *The percentage of regions in a MC whose GTA is the DA (at a family level)*

%DAC (of a MC) - % Dominant Architecture (Clan). *The percentage of regions in a MC whose GTA is the DA (at a clan level)*

%DACF (of a MC) - % Dominant Architecture (Clan) and Fewer. *The percentage of regions in a MC whose GTA is either the DA (at a clan level) or a subset, including unannotated regions.*

%DACFA (of a MC) - % Dominant Architecture (Clan), Fewer and Additional. *The percentage of regions in a MC whose GTA is either the DA (at a clan level), a subset or a superset.*

%UNK (of a MC) - % Unannotated protein regions. *The percentage of regions in a MC not overlapping with any Pfam region, therefore to be considered as unannotated by Pfam.*

O_{MC} - Average overlap of a MC regions with its DA. *DA is considered either at a family or ad a clan level; this quantity is computed on the subset of common* *regions.*

PF - a Pfam Family. *In particular, a Pfam family whose member regions are restricted to those found in the working database, namely UniRef50 v. 2017_07.*

%MC$_i$ (of a PF) - % of protein sequences of a PF found also in metacluster MC$_i$. *The percentage of regions in a MC whose GTA is the DA (at a family level)*

O_{PF,MC_i} - Average overlap of a PF's regions with a metacluster's (MC$_i$) regions. *This quantity is computed on the subset of common regions.*

DMC (of a PF) - Dominant metacluster. *The MC that maximises* O_{PF,MC_i} *provided that* $\%MC_i > 50$

%DMC (of a PF) - Percentage of DMC regions. *%MC*$_i$ *for MC*$_i$*=DMC.*

Introduction

Characterizing the function of proteins is an extremely important step in understanding biological processes. This characterization can be performed, in principle, by dedicated experiments, both in a wet lab or through computational simulations - which are overall expensive. The number of proteins that have been characterized with such methods is still relatively small. Instead, in the last decades the number of protein sequences known has grown exponentially, especially thanks to High Throughput Genomic sequencing experiments. Proteins that are known only at a sequence level outnumber those with an experimental characterization by orders of magnitude, and we expect this gap to increase over the next years.

Proteins are polymers constituted of amino acidic monomers. The shapes they take, the functions they perform and their interactions with other molecules depend on their amino acidic sequence. Many successful and popular tools in bioinformatics are based on the same key idea: if a group of proteins share similar amino acidic sequences, they will have the same structural fold, and may share similar functions.

A family is typically defined as an ensemble of protein regions that are evolutionary related (homologous). Homologous regions need not span the entire length of any sequence. In fact, many proteins are covered by multiple families which they share independently with several other proteins. In particular, two proteins A and B can have sequences that are homologous only in one of their regions, and can be totally unrelated over the remainder of the sequence. The region of protein A which is unrelated to protein B can in turn be homologous to a region of a third protein, C. The set of protein families that characterize a protein is what we call its ”family architecture”. Nature seems to use families as Lego bricks, by combining families of different shapes and functions in different architectures, to obtain more complex shapes and functions in whole proteins [1].

Many protein family databases exist, such as Pfam [2], SUPERFAMILY [3], Gene3D [4], ECOD [5] or InterPro [6]. These databases have been built over many years, often with significant manual work for their curation. Protein families defined in these databases are very reliable, and allow to classify a vast quantity of protein sequences. Manual curation, however, can be a rather slow process and with the rapid increase of protein sequences that are being deposited into public databases such as UniProtKB, large areas of the sequence space remain without family annotation.

In this thesis we describe an algorithm for defining protein families which is totally unsupervised and based only on sequence information. We are not the first attempting to develop an approach based on these premises. For example, until 2015 Pfam used the ADDA [7] algorithm to automatically generate Pfam-B, a complementary database of Pfam-A, its human-curated collection of families. Similarly the EVEREST algorithm [8] defines protein families from sequence data, through pairwise alignments and using a machine learning method to infer the concept of ”family” from Pfam. Our approach takes a different path from these former contributions, and it has been developed to be able to deal with the large size of current protein sequence databases. It is an attempt to port in the field of protein annotation Density Peak Clustering [9], a recently developed algorithm which has been successfully used for unsupervised classification in many different fields. The protocol clusters together local pairwise alignments using Density Peak Clustering (DPC) in a two-step procedure. Given a set of proteins to analyse (the ”query set”), we first search for local pairwise alignments on a large database of sequences, UniRef50 [10], or the “search set”. Next, independently for each query sequence, we identify all regions that align to sequences in the search set. Alignments between a query sequence and sequences in the search datasets are typically many thousands, sharing different degrees of overlap on the query. We cluster them by DPC, obtaining what we call ”primary clusters”, which provide a first approximation of the architecture of the query sequence: each cluster, potentially, belong to a separate family. However, the information derived from this first clustering is extremely specific to the query sequence. To reduce this specificity, primary clusters of different query sequences are grouped into ”metaclusters” (MCs), based on the number of search sequence regions they have in common. This step is performed once again with DPC: clusters of regions obtained in this way are named ”metaclusters”. Each MC is then what we define as a potential seed for a protein family. We call our new method ”DPCfam”, and it is described in Section 2.

We first use DPCfam on two small sets of query sequences, each containing sequences in a single Pfam clan (PUA [11] and P53-like [12], respectively), where the Pfam database [2] is what we use as ground truth to compare our results with.

This part of the thesis serves as a proof-of-principle experiment. We wrote a first implementation of DPCfam (described in Section 2.3), and we performed an in-depth analysis of the metaclusters it produced (Section 3.2). Most of the Pfam families of the query sets are found also as metaclusters (MCs), together with some apparently inconsistent MC. In this last case, we perform an in-depth analysis of the clustered protein regions using structural data (when available), sequence analysis or profile-profile comparisons. This allows us to better understand the reasons behind DPCfam's alternative family annotation of these regions. Interestingly, in a few cases, we are able to propose improvements to the Pfam classification in terms of family boundaries, clan membership and existence of new families. This analysis has been submitted as a paper to "BMC bioinfomatics" (currently in review).

The main novel contribution presented in this thesis is an all-to-all clustering of the protein database of reference, UniRef50. Such a task is computationally expensive, and to effectively implement the DPCfam protocol required a careful optimization and parallelization of the code. In Section 2.4 we describe this implementation, with a focus on the measures we took to speed up the method's running time. The clustering approach handles data volumes of the order of 5 TeraBytes. To give an idea of the difficulty of the computational problem that we had to solve, the matrix containing the distances between all the primary clusters (necessary to build metaclusters) has $25 \cdot 10^9$ non-zero entries. Its computation took around 30,000 CPU hours. The computational pipeline has been designed in order to make it recursively updatable. This allows reducing notably the computing time required to build incremental versions of DPCfam MCs.

From the all-to-all clustering of UniRef50 we obtain about 45,000 metaclusters, bound to contain more than 50 protein regions and to have an average amino-acidic length larger than 50. While replicating on all of these families the in-depth analysis done for the proof-of-principle experiments is outside of the scope of this work, we are still able to apply to them a lot of the concepts we developed there especially for what concerns the comparison with the Pfam gold standard (Section 3.3.1). The Pfam database counts around 18,000 families: we restricted the comparison to those that have more than 100 member regions in proteins that belong to our reference sequence database UniRef50. In 80% of the cases at least half of the family members are found as members of a same MC, not necessarily with the same boundaries; but in 30% of the cases, also the family boundaries are very well replicated by the DPCfam classification. We recall that a significant fraction of families in the Pfam database are defined (or refined) by taking advantage of additional information besides sequence, including structure and function of their members, when available. Although this information is encoded in the sequence of proteins, there is no guarantee that it can be consistently, meaningfully extracted from local sequence alignments: we therefore believe that the agreement that we find between the manually-curated Pfam classification and the automatically-generated DPCfam classification, though partial, constitutes a non-trivial and thus encouraging result.

In addition, our protocol finds many thousands more potential families than those found in Pfam. While it is true that redundancies are possible (still, the most evident are reduced by the merging step of DPCfam, described in Section 2.2.2.3), there are more than 10,000 metaclusters whose regions have no Pfam annotation and which are candidate new protein families. The clustering protocol we developed can then both assist in refining family annotation (when families are already known) and be used to define possible novel families. Results obtained so far will be made available to the scientific community in a dedicated repository.

The Thesis is organized as follows:

- In Chapter 1 we present an introduction on protein families, focusing on the homology relationships between protein regions of the same family. We subsequently introduce the main theory needed for protein sequence analysis, including Local Pairwise Alignments and Hidden Markov Models. Finally, we present the most relevant protein sequence databases, protein families databases, and we describe some of the existing methods for automatic protein family classification.

- In Chapter 2 we introduce our clustering method. First we describe the Density Peak Custering algorithm, and the automatization procedure we introduced. Then, we describe the general algorithm, named DPCfam. Finally we describe the two implementations we developed: DPCfam0, used for the small, proof-of-concept analysis, and the DPCfam optimized version, tailored to the HPC facilities at our disposal.

- In Chapter 3 we present the results obtained from the two implementations of DPCfam. We first present the measures we developed to analyze metaclusters, with particular attention on comparing MCs to Pfam families. In Section 3.2

we show the results obtained on two small datasets of proteins (PUA_UR50 and P53_UR50), namely the proof-of-principle experiment: this includes an in-depth analysis of the metaclusters, including making large use of structural data. In Section 3.3 we show results obtained clustering all UniRef50 protein database.

- In Chapter 4 we present the conclusions and discuss the future perspectives of the method.

Chapter 1

Protein families

Proteins are essential components of any living organism: with no metaphor they can be described as nanomachines, each of them devoted to specific jobs in the complex organizational network of a cell. We are allowed to think of them as the main component that makes an organism *function*: thus, the necessity to study, characterize and classify proteins is evident.

The Central Dogma of Molecular Biology [13] tells us that biological information flows from DNA to RNA, and from RNA to proteins; a similar concept has been developed for *Structural Biology*: in this case, information flows from the protein sequence to the protein structure, and from structure to function [14]. In the last decades both dogmas have been somehow invalidated by the discovery of functional RNA and of natively unstructured proteins, but both are still useful guides in understanding the protein world. In our bravest dreams of reductionist physicist, we would like to be able to completely infer any protein's characteristic just by knowing the DNA sequence that produced it, by following the information flow of the two dogmas: DNA make RNA, which makes protein sequence; protein sequence determines structure, which determines function. By limiting ourselves to the Central Dogma of Structural Biology, we can re-shape our purpose: we want to infer the characteristics of any protein from its amino-acidic sequence (protein sequence determines structure, which determines function... "more or less"). Such a purpose becomes more and more meaningful if we consider that in the last years we have witnessed an exponential growth of protein sequences deposited in public databases, covering a large variety of organisms. The largest public database for protein sequences (UniRef database of UniprotKB [10]) counts today (v. 2020_05) 242,399,302 total entries, and it more than doubled from November 2016 - the month when I started my PhD.

What to do with all this information that has been and is constantly being collected? How to exploit at best the relationship between a protein's amino-acidic sequence and its structure and function?

Here protein families enter the game.

The concept of *protein family* has been a key for understanding the protein world in the last decades [1] [15] [16]. The next sections provide a review on protein families and on all the concepts and instruments involved in their definition. We will then give a brief description of some protein sequence databases, with a particular attention to UniprotKB [10]. We will also describe Pfam [2], a well-known database of protein families. Both UniprotKB and Pfam have been widely used in this thesis.

1.1 Families and Domains

As a general description of a protein family, we can say that: a ”protein family” is a set of protein [1] modules (or protein regions) derived from a common ancestor. Members of the same protein families share the same three-dimensional structure (if any), and may share similar functions.

A key concept in understanding protein families is their *modularity*. Protein families do not necessarily span whole proteins, but very often only protein regions. Usually, but not always, an evolutionarily conserved protein region (or module) is contiguous though the amino-acidic sequence. Moreover, these modules can appear in the same protein, in succession (named *repeats*), or together with other modules, arranged in different orders, namely with different *architectures*.

At the basis of the definition of protein families lies *evolution*. Proteins - in particular, their amino-acidic sequence - are the result of evolution: this affects strongly protein families. Since member regions of a family originate from a common ancestor, their sequences show evolutionary patterns. For example, studying the relationships between members of the same family one can infer phylogenetic trees. These evolutionary patterns are strongly constrained: modules of the same family fold in the same overall three-dimensional structure, and the preservation of such fold imposes a constraint on the amino-acidic sequence that a member of the family can take. This property can be used backwards: from a set of amino-acidic sequences of the same family, one can infer the structural constraints, for example using Direct Coupling Analysis [17], extracting structural information from sequence information.

[1] When writing about ”proteins” specifically refer to single chains, or amino-acidic sequences, as most of the literature in bioinformatics does.

While we can say that members of the same protein family share the same fold, their functional relationship is less direct. In general, the slightest change in amino-acidic sequence can change the function of a protein. For example, the bonding affinity of a protein for a ligand can notably change with a single amino-acidic substitution, while the same substitution would not affect at all the protein's fold. Still, oftentimes proteins within the same family share functions that are similar: if we know the function of a member of the family, we can hope to infer important information about the function of the rest of the family members.

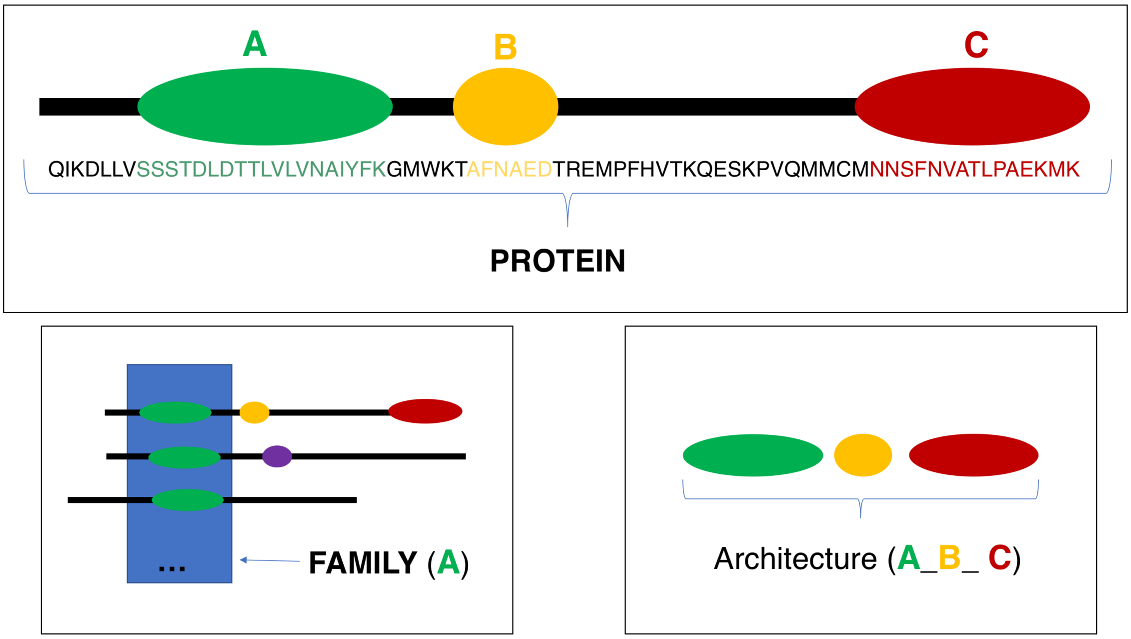

A
B
C
QIKDLLVSSSTDLDTTLVLVNAIYFKGMWKTAFNAEDTREMPFHVTKQESKPVQMMCMNNSFNVATLPAEKMK
PROTEIN
...
FAMILY (A)
Architecture (A_B_ C)

Figure 1.1: Schematic organization of protein families as evolutionary modules. *Top Panel*: schematic representation of a protein as a sequence, with three evolutionary modules of three different families (A,B and C). *Bottom Left Panel*: a set of similar modules, appearing in different proteins, is a protein family. *Bottom Right Panel*: the ordered set of different module in a protein (A, B and C, cfr. Top Panel) defines che protein's architecture (A_B_C)

To summarize, finding, cataloguing and describing families of evolutionary related (homologous) protein regions can provide important information for annotating these regions' structure and function. Crucially, traces of the evolutionary relationships between members of the same family can generally be found in their sequences. In fact, sequence similarity is the most widely measure used to group protein sequences into families.

In the next sections we will present some of the main methods and instruments that are used to identify sequence similarities, and how they are used in some well-known protein family databases, such as Pfam.

1.2 Sequence Alignments

Amino-acidic sequences (proteins) are made from 20 standard amino-acids, which are conventionally named using letters from the Latin alphabet: therefore we can think of them as a string of letters. Length of such strings can vary from few dozens to several thousands of letters. Proteins are under evolutionary constraint: changes in the DNA coding sequences can affect proteins' sequence. There are three main mutations in sequences: the substitution of an amino-acid with another one, the deletion of an amino-acid from the sequence, and the insertion of a novel amino-acid in the sequence.

1.2.1 Pairwise Alignments

A pairwise sequence alignment compares two protein sequences, trying to understand if they are biologically related. If related, they should share a common ancestor: but such ancestor is unknown and cannot be used in the comparison. Still, we can compare directly the two sequences, and try to recollect the most probable set of mutations that can transform one sequence in the other one. In doing to, we "align" the two sequences.

Let us consider the simplest case: we want to align two sequences which differ only in one amino-acid. An alignment of this type is the following:

```
ACDKDC
| ||||
AGDKDC
```

Here lines represent the exact matches (being the mutation here in the second amino-acid). As another simple case, we want to align two sequences with a deletion/insetion. For example:

```
D-KLP
| |||
DGKLP
```

The dash in the first sequence represent a "gap". A gap can be both interpreted as a deletion or as a an insertion event. As the number of mutations between one sequence and the other one increases, the process of aligning becomes more and more difficult. However, this task can be performed by a deterministic algorithm, namely the Needleman-Wunsch algorithm [18]. This is dynamic programming algorithm that is computationally expensive but guarantees to find the best global pairwise alignment(s) between two sequences. To obtain the "best" pairwise alignment, we need a scoring system. There are two kind of events to score - amino-acid substitution and insertion/deletion (or gap opening). In the first case, substitution matrices are used: for each couple of amino-acids, the matrix gives a score, associated to the probability to observe the substitution from amino-acid (a.a.) i to a.a. j in homologous proteins. The more probable the substitution, the better the scoring (and vice-versa). A large number of these substitution matrices has been defined using different strategies, statistical instruments, and datasets of reference; the most famous are BLOSUM matrices [19] and PAM matrices [20]. Different matrices are devoted to align sequences whose homology depends on different evolutionary times: some are more effective in detecting close relationships (short evolutionary time, e.g. BLOSUM80), other in detecting mid and far relationships (longer evolutionary times, e.g. BLOSUM62 and BLOSUM45).

In the second case (gap opening), the insertion/deletion event can be modeled in several ways, but the most used is the "affine gap penalty model" [21]. Here we have "gap opening" and a "gap extension" contribution: the first one takes into account the cost of generating a gap in a sequence; the second one accounts for the gap's length. Thus, if we consider the case

```
D--LP
|  ||
DGKLP
```

we must take into account both the the "opening" contribution, scoring a certain penalty (G), and then the elongation contribution, counting for the two gaps each with a penalty L: as a result, this two-dashes gap will cost $G+2L$. In general, a gap of length n will count a $G+nL$ penalty. Values of G and L are usually empirical, and can vary depending on the purpose of the alignment.

1.2.2 Local Pairwise Alignments

One, instead of aligning entire proteins, can be willing to align subsections of proteins. For local alignments, the counterpart of the previously cited Needleman-

Wunsch algorithm is the Smith–Waterman algorithm [22], which is in turn a deterministic dynamic programming algorithm. It can be computationally expensive, but guarantees to find the best scoring local alignment(s). Also in this case scoring matrices and gap penalties are used.

Local pairwise alignments are those of our interest, being capable to find local homologies in protein sequences, which are at the basis of the definition of protein families. As we will see further, in this thesis we use huge amounts of local pairwise alignments to identify potential families in a protein dataset. We start with a given protein (*query*), and we search for all meaningful local alignments that we can find on any other protein (*searches*). Intuitively, finding several local alignments on the same region of a query protein signals the presence of a possible evolutionary conserved region, namely a protein region that is part of a family.

To do this, we must first know what do we mean by meaningful alignment.

1.2.3 Assessing homology between alignments: E-Value

The scoring system we described is used to find, among all those possible, the best pairwise alignment between two sequences: however, even the best scoring alignment may be a nonsense alignment. Let us consider the case where we want to align the sequence DKP to AAITESDKFFINGSNW: at a glance, we can guess the best alignment between these two sequences to be

```
------DKP-------
      ||.
AAITESDKFFINGSNW
```

for a global alignment, and (maybe)

```
DKP
||.
DKF
```

for a local alignment. Here we signaled with the ”.” the presence of a good (high scoring) substitution between P and F.

While the global alignment will have a very low score, the local alignment may score fine enough. But this alignment has another ”property”: the first sequence (DKP) has been obtained typing randomly on the keyword; while the second sequence is a chunk of a real protein. Does such an alignment make sense? Not much. Indeed, it is very easy to find local alignments with very small sets of amino-acids. But it is also possible with longer strings, especially if the database we are searching is very large. When we want to assess if an alignment may actually represent a homology relationship between two sequences, we can ask ourselves what was the chances that such a match happened randomly. In particular, what was the probability to find this match in the case one of the two sequences was randomly generated? This is measured by the E-Value [23][24]. It is a proxy of a probability measure: small E-Values tell us that it was improbable to generate randomly a sequence that produced exactly this match, and thus fosters the hypothesis that the two aligned sequences are homologous. Karlin and Altschul [24] defined the E-Value of a pairwise alignment as:

$$E = Kmne^{-\lambda S} \tag{1.1}$$

where S is the alignment's score, m is the number of amino-acids in a *query* sequence, while n is the total number of amino-acids in the protein database where we are searching. The λ parameter is used to normalize scores, and it depends on the substitution matrix used; K can be thought as another normalization factor, which takes in consideration the correlation between local alignments starting at very close positions [23], therefore it represent a normalization factor based on the properties of the protein sequence space searched.

We note that the concept of the E-Value is asymmetric: it considers one of the two sequences to be possibly random (the *search* sequence), not the other one. Moreover, the concept of "random sequence" is not based on a purely random sequence generator, but on the substitution matrix used and on its statistical properties. Finally, when considering the probability to find randomly an alignment, the E-Value takes into account the size of the proteins' set we are searching for local alignments.

1.2.4 Percent Identity

Being often used in this work, we describe briefly the Percent Identity (PI) of an alignment. Given the two aligned sequences, for each site one can check if the two contain exactly the same amino-acid: doing this for every site will give a measure of how much the two sequences are "identical". 100 PI is achieved with two identical sequences (in global alignment), or when one sequence is exactly a sub-sequence of the other one (in local alignments). We note that alignments between two protein regions of the same family usually score below 50 PI (most commonly around 30

PI). Indeed, families contains sequences that during evolution diverged significantly.

1.2.5 BLAST

Smith-Waterman [22] algorithm is too computationally expensive to perform a large quantity of local pairwise alignments; in this case heuristic algorithms are preferred. These algorithms do not find the perfect scoring alignments between two sequences, but are much faster. "Good enough" alignments found in this way are still very useful, and heuristic algorithms are the most used in bioinformatics. Of these, a "gold standard" for local pairwise alignment is BLAST [25] [26], "Basic Local Alignment Search Tool".

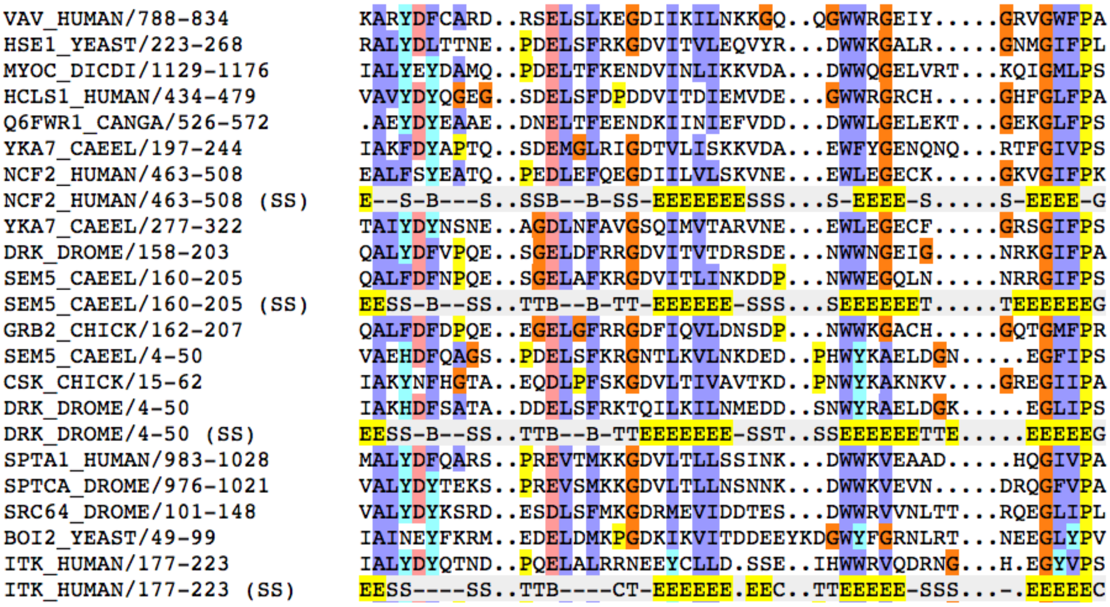
BLAST searches for local pairwise alignments on a query sequence scanning a protein sequence database. As a result, BLAST finds local alignments between a query sequence and a search sequence found in the database, within a certain score or E-Value. In fact, the *query* and *search* terminology we used so far is borrowed from BLAST's vocabulary. Also the concept of E-Value [23] [24] has been developed initially by the BLAST creator himself, Stephen F. Altschul, testifying the importance that this algorithm has in bioinformatics.

BLAST's search strategy is based on searching in the database for matches between small "words" (3-4 amino-acids) contained in the query and small words contained in the search sequences. Searching by small words is an easy task because of their reduced statistics. The program compiles matching tables that fasten the search. Once it finds a significant match between small words, it extends the alignment in both directions, adding pairs of amino acids. The extension is stopped once the score drops under a certain value: in this way, two sub-sequences (one of the query, one of the search sequence) are found, and an optimal alignment between them is computed: this will be the resulting local alignment (one of the many) reported by BLAST.

Several implementations and varied versions of this algorithm exists; in this thesis we will use the NCBI BLAST+ suite [27], and in particular the blastp program, which is specific for protein-protein alignment. The standard scoring matrix used by blastp is BLOSUM62.

1.2.6 Multiple Sequence Alignments

As one can align two sequences, the same can be done for a larger set of sequences. This task is much more complicated than the previous one. [28].



```
VAV_HUMAN/788-834            KARYDFCARD..RSELSLKEGDIIKILNKKGQ..QGWWRGEIY.....GRVGWFPA
HSE1_YEAST/223-268           RALYDLTTNE..PDELSFRKGDVITVLEQVYR...DWWKGALR.....GNMGIFPL
MYOC_DICDI/1129-1176         IALYEYDAMQ..PDELTFKENDVINLIKKVDA...DWWQGELVRT...KQIGMLPS
HCLS1_HUMAN/434-479          VAVYDYQGEG..SDELSFDPDDVITDIEMVDE...GWWRGRCH.....GHFGLFPA
Q6FWR1_CANGA/526-572         .AEYDYEAAE..DNELTFEENDKIINIEFVDD...DWWLGELEKT...GEKGLFPS
YKA7_CAEEL/197-244           IAKFDYAPTQ..SDEMGLRIGDTVLISKKVDA...EWFYGENQNQ...RTFGIVPS
NCF2_HUMAN/463-508           EALFSYEATQ..PEDLEFQEGDIILVLSKVNE...EWLEGECK.....GKVGIFPK
NCF2_HUMAN/463-508 (SS)      E--S-B---S..SSB--B-SS-EEEEEEESSS...S-EEEE-S.....S-EEEE-G
YKA7_CAEEL/277-322           TAIYDYNSNE..AGDLNFAVGSQIMVTARVNE...EWLEGECF.....GRSGIFPS
DRK_DROME/158-203            QALYDFVPQE..SGELDFRRGDVITVTDRSDE...NWWNGEIG.....NRKGIFPA
SEM5_CAEEL/160-205           QALFDFNPQE..SGELAFKRGDVITLINKDDP...NWWEGQLN.....NRRGIFPS
SEM5_CAEEL/160-205 (SS)      EESS-B--SS..TTB--B-TT-EEEEEE-SSS...SEEEEEET.....TEEEEEEG
GRB2_CHICK/162-207           QALFDFDPQE..EGELGFRRGDFIQVLDNSDP...NWWKGACH.....GQTGMFPR
SEM5_CAEEL/4-50              VAEHDFQAGS..PDELSFKRGNTLKVLNKDED..PHWYKAELDGN.....EGFIPS
CSK_CHICK/15-62              IAKYNFHGTA..EQDLPFSKGDVLTIVAVTKD..PNWYKAKNKV....GREGIIPA
DRK_DROME/4-50               IAKHDFSATA..DDELSFRKTQILKILNMEDD..SNWYRAELDGK.....EGLIPS
DRK_DROME/4-50 (SS)          EESS-B--SS..TTB--B-TTEEEEEEE-SST..SSEEEEEETTE.....EEEEEG
SPTA1_HUMAN/983-1028         MALYDFQARS..PREVTMKKGDVLTLLSSINK...DWWKVEAAD.....HQGIVPA
SPTCA_DROME/976-1021         VALYDYTEKS..PREVSMKKGDVLTLLNSNNK...DWWKVEVN.....DRQGFVPA
SRC64_DROME/101-148          VALYDYKSRD..ESDLSFMKGDRMEVIDDTES...DWWRVVNLTT...RQEGLIPL
BOI2_YEAST/49-99             IAINEYFKRM..EDELDMKPGDKIKVITDDEEYKDGWYFGRNLRT...NEEGLYPV
ITK_HUMAN/177-223            IALYDYQTND..PQELALRRNEEYCLLD.SSE..IHWWRVQDRNG...H.EGYVPS
ITK_HUMAN/177-223 (SS)       EESS----SS..TTB----CT-EEEEEE.EEC..TTEEEEE-SSS...-.EEEEEC
```

Figure 1.2: Example of a MSA. First 23 rows (of 55) of the SH3_1 Pfam family seed. Some columns are colored to show the persistence of a certain kind of amino-acid over the sequences in the same position, e.g. dark blue highlights small or hydrophobic amino-acids (C,A,V,L,I,M,F,W).

In building MSAs, the general goal is to collect on the same column those amino-acids that, belonging to homologous proteins, occupy the same three-dimensional position in the proteins' fold. The existence of a homology relationship between the elements of the MSA is of extreme importance, being the basic assumption on which the MSA is built. If we believe that all the sequences we are aligning are evolutionary related, we are allowed to search for conserved - or almost conserved - regions, and to interpret a MSA's column as a set of amino-acids that originated from the same ancestor, diverging across evolution. MSAs are one of the objects used to represent protein families, possibly the older one. As we will see, most of the protein family databases are based on manually-curated MSAs representing protein families.

Implementing a reasonable scoring system for MSAs is not straightforward. Hand-made MSAs, built by experts with a trained eye, are often more reliable than those automatically generated searching for the optimal alignment by an unsupervised approach. It is not uncommon for bioinformaticians to build MSAs starting from automatically generated ones, and then to correct them manually. In the last decades several algorithms and programs to generate automatically MSAs have been proposed: while dynamic programming methods are possible, heuristic ones are more feasible. As examples we can cite T-COFFEE [29], Clustal Omega [30] and MUSCLE [31].

1.3 Modeling families as profile-HMM

We said that protein families can be represented as MSAs; from these, one can build a compact representation called profile-HMMs[32]. A profile-HMM of a protein family is based on a Hidden Markov Model, and it aims to give a statistical representation of the family using information from its MSA.

For example, given a column (or site) of a MSA, we can compute the probability for a certain amino-acid to appear, building something similar to a scoring matrix (specifically a PSSM, Position Specific Scoring Matrix). This is easily done in MSAs (or MSAs' regions) with no gaps, but MSAs of protein families usually contain a vast quantity of gaps, including a number of regions which are mostly gaps: it is then necessary to model gaps too.

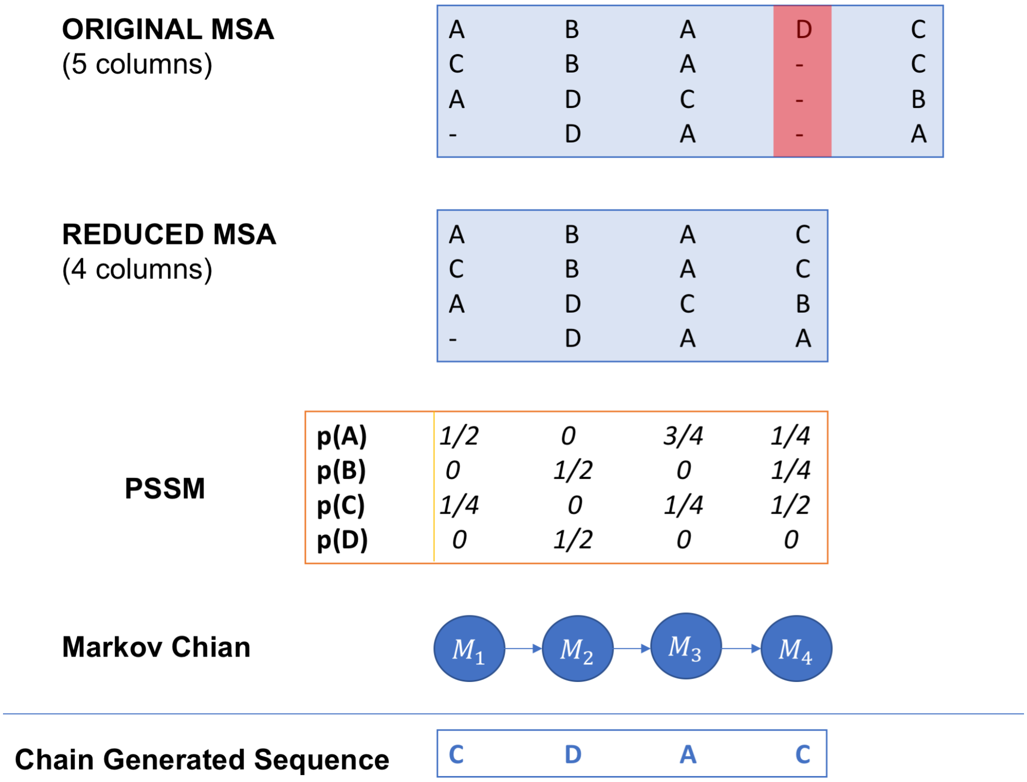
Profile-HMMs provide a probabilistic representation of a family. In principle profile-HMMs are generative models, capable to generate new sequences of the family they represent; such a process can be used to score the probability for a given sequence to be produced by the profile-HMM: if this is the case, the sequence found as a "hit" can be considered as a family member. To assess the quality of a hit one uses the E-value.

HMMER [33] [34] is one of the most used software to generate profile-HMMs and to use them to search for hits in a protein database.

1.3.1 A brief overview of profile-HMMs

Given a family MSA of C columns, a first step to generate its profile-HMM is to compute its PSSM. Since PSSM does not take into account gaps or insertions, usually profile-HMM generating algorithms remove from the MSA all columns containing an excessive number of dashes, namely columns where gaps appear with a frequency larger than a threshold θ. Then, from a reduced MSA containing c columns, the PSSM is computed on the remaining columns, ignoring dashes when they occur.

Subsequently, a first Markov Chain [35] is built: this is a linear sequence of states M_i, one for each column. In this particular chain, the probability to jump from one state M_i to the subsequent M_{i+1} is 1 (corresponding to moving through the sequence): once in state i, the probability to emit a certain amino-acid is defined by the previously computes PSSM. This model can generate sequences following the PSSM distribution. These sequences are ungapped, therefore they always have the same length (c) (see Figure 1.3).



A	B	A	D	C
C	B	A	-	C
A	D	C	-	B
-	D	A	-	A

A	B	A	C
C	B	A	C
A	D	C	B
-	D	A	A

p(A)	1/2	0	3/4	1/4
p(B)	0	1/2	0	1/4
p(C)	1/4	0	1/4	1/2
p(D)	0	1/2	0	0

C	D	A	C

Figure 1.3: Graphical representation of the first steps to build a profile-HMM from a MSA based on a toy-model MSA. First, the MSA is reduced to significant columns, from which a PSSM is inferred. Then a Markov Chain is created, so that it can generate ungapped sequences according to the given PSSM.

The second step is to add to this simple Markov Chain a set of hidden states, which will transform it in a Hidden Markov Chain (or Hidden Markov Model). Such hidden states model insertion and deletion events. For example, from any state M_i it is possible to move to state I_i, which will result in adding a gap between the amino-acids emitted by M_i and M_{i+1}. However, we must account for the possibility to have gaps longer than a single site: this is obtained by moving from state I_i to I_i itself, adding dashes to the sequence being generated. Deletion states, on the other hand, account for the possibility to jump directly from site M_i to M_{i+n}, where n is the length of deletions; therefore, once in state M_i it is possible to jump on the deletion state D_{i+1}, from which again it is possible to go back to M_{i+2} (skipping site M_{i+1}) or to continue to delete, moving to state D_{i+2}. Finally, to coordinate all these states, a Start state and an End state are added at the beginning and at the end of the Markov chain. The graph representing states and transition states is represented in Figure 1.4

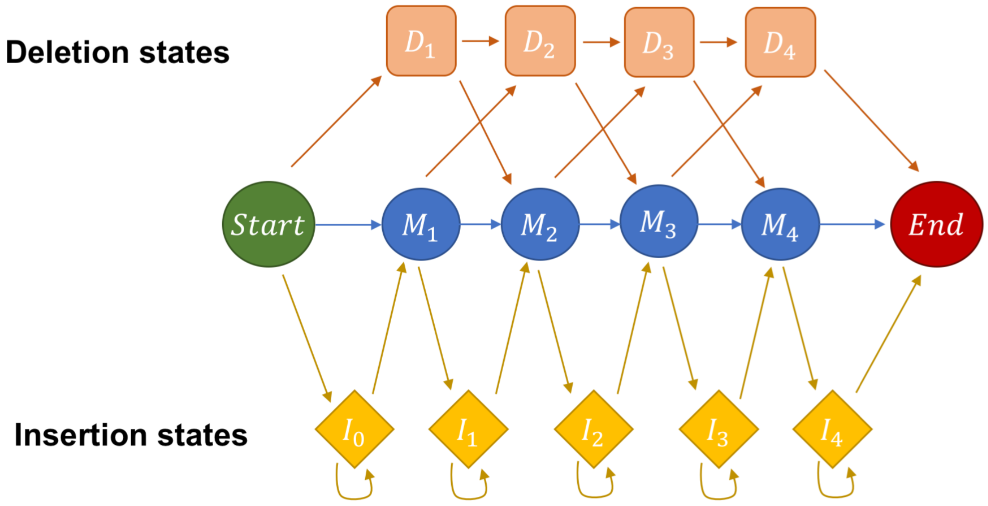


Figure 1.4: Graphical representation of a profile-HMM's states, including amino-acid emission states (M_i), insertion states (I_i), deletion states (D_i), and Start/End states.

The probability to move between sequence states (M_i), insertion states (I_i) and deletion states (D_i) is usually computed using the Viterbi algorithm for Markov Models (see [33] for further details).

1.4 Protein databases

In this section we will give a brief overview of the main protein databases freely available to the scientific community. We will first focus on the Uniprot's protein sequence databases, and then we will introduce the Protein Data Bank (PDB), which is the largest public repository of protein structures.

1.4.1 Protein Sequence Databases: Uniprot

The Uniprot database [10] can be considered the largest and most comprehensive database for protein sequences. The main database, named The UniProt Knowledgebase or UniProtKB, is the sum of two historical sequence databases: Swiss-Prot and

TrEMBL [36]. The first one contains ”curated” entries, namely protein sequences that have been assessed at an experimental level; the second one is automatically generated, containing proteins inferred from genomic data. The two databases grow at a very different pace (see Figure 1.5); currently Swiss-Prot counts about 550,000 entries, while TrEMBL around 195,000,000.

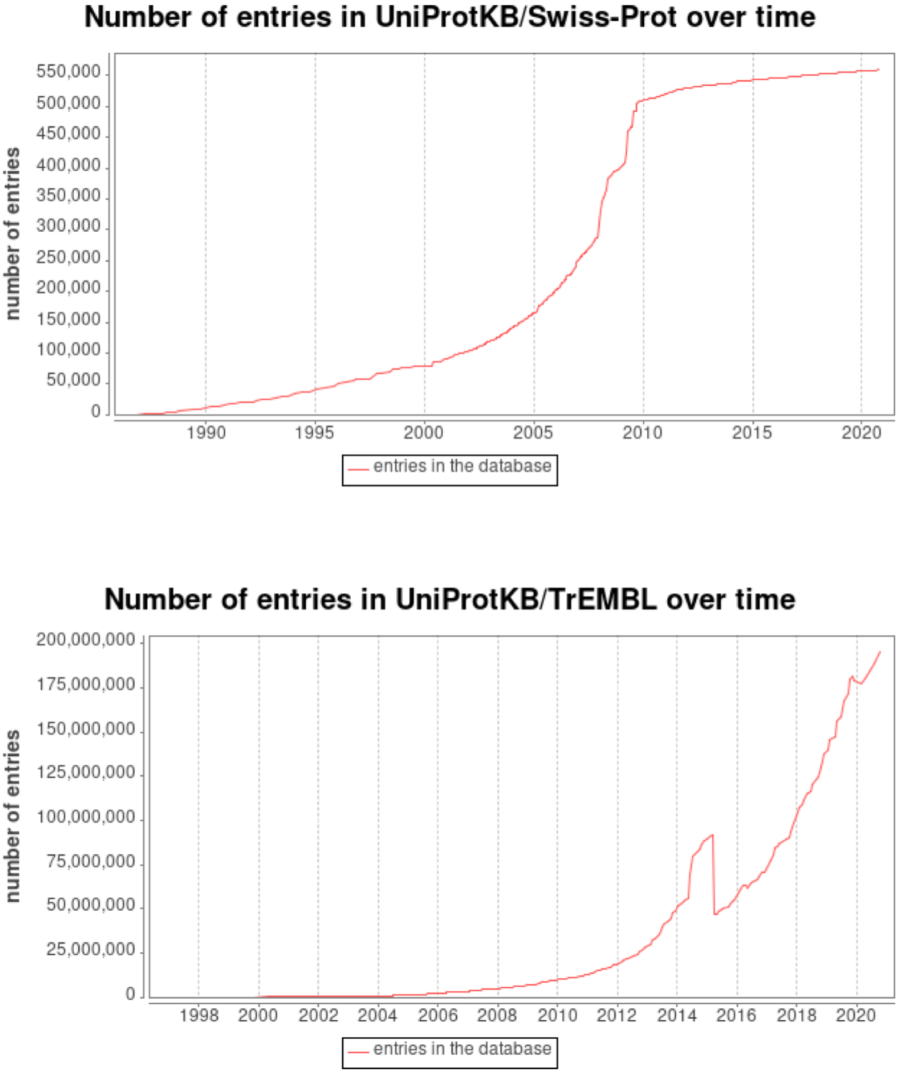


Figure 1.5: UniProtKB (Swiss-Prot and TrEMBL) statistics over time, according to the UniProt webiste (https://www.uniprot.org/) (Updated at version 2020_05)

1.4.1.1 UniRef

UniRef is a set of non-redundant databases defined by the UniProt consortium. The largest (and most redundant) database, named UniRef100, contains protein sequences form UniProtKB and some selected entries of another database, UniParc. Sequences with 100 Percent Identity (P.I.) are aggregated, and a single sequence is used as a representative. UniRef90 is built from UniRef100 reducing the redundancy between sequences to 90 P.I.: sequences with more than 90 P.I. are clustered together, using CD-HIT [37], and only one representative among them is chosen. Subsequently, from UniRef90 is derived the UniRef50 database, using again CD-HIT so that sequences with more than 50 P.I. are clustered together. The UniRef50 database offers a good representation of the known protein sequence space at a low-redundancy level.

1.4.1.2 Reference Proteomes

Reference Proteomes are sets of proteins derived from organism the genome of which has been fully sequenced. This protein sequence dataset aims to provide broad coverage of the tree of life. While this is commonly considered as a non-redundant database, it may contain multiple proteome versions for species of particular relevance.

1.4.2 Protein Structure Databases: PDB

The Protein Data Bank (PDB) [38] is a protein structure database containing the largest set of publicly available three-dimensional protein structures. These have been solved using X-Ray crystallography, NMR or other techniques. We note that normally PDB entries do not contain full-length proteins, but rather chunks of proteins, often in contact with each other. The different objects captured in a PDB entry are named chains, usually labeled with capitals letters (A, B... etc.). Proteins in PDB entries are usually part of the Swiss-Prot database.

1.5 Protein Family Databases

Several resources have been developed to identify and catalog protein families. Usually, family databases use different definitions to build a *seed* Multiple Sequence Alignment (MSA) for each of their families. In many cases such MSAs are used to search for other members of the family in a protein sequence database: results obtained as such are collected in the actual definition of the given protein family. There are several ways to perform this search: here we will focus on databases that generate profile-HMMs (see Section 1.3) of seeds' MSAs, and then use them to search for other members of the family.

Family databases can be distinguished between those which build seeds using structural information and those which use more sequence-based information. Additionally, we can distinguish between databases using human-curated information and automatically generated databases: in the protein family databases panorama, the first ones are dominant. Finally, there are cross-referencing resources (such as InterPro [6]) combining data derived from different databases.

1.5.1 Structure-based Protein Family Databases

Structure-based Protein Family databases, such as CATH-Gene3D [4], SUPERFAMILY [3], or ECOD [5], build seeds starting from structure-based MSAs. For example, the CATH database contains structural families, built starting from structural data: a mixture of automatic and manual methods are used to build CATH's families and the respective MSAs. Profile-HMMs of CATH's families are used by Gene3D to scan through a large protein database, increasing the set of members of each family. Similarly, SUPERFAMILY uses as seeds SCOP's [39] structural families. Both Gene3D and SUPEFAMILY databases use iterative profile building. A more recent database, named ECOD, builds its own structure-based seeds and associated families.

These databases, the seeds of which are based on structural data, usually offer high-quality definition of protein families, detecting far homologies. On the other hand, building seeds starting from structure data limits the protein sequence space regions that can be explored by their profile-HMMs.

1.5.2 Sequence-based Protein Family Databases: Pfam

Sequence-based databases of protein families aim to classify the protein sequence space widely; here we will focus on the Pfam database [2]. Pfam's seeds are built using a variety of sources of information. For example, at the very beginning of Pfam's history (1997 [40]), Pfam's seeds were built using information from other existing domain databases, articles, structural data, BLAST results and other kind of databases such as repeats databases. Protein families originated from these kind of information constitute the Pfam-A family database. During Pfam's history, several automatic tools (such as Domainer [41] or ADDA [7]) have been used to build a complement to the main database, named Pfam-B. This second database contained automatically-generated families which were gradually integrated in the Pfam-A collection using manual curation. The origin of Pfam's families is then extremely heterogeneous.

Pfam offers as a family description two objects: the seed's profile-HMM and the MSA of the "full alignment". This last MSA is the result of the respective seed's search through a non-redundant database (usually the Reference Proteomes), and according to Pfam it constitutes the family definition.

Currently Pfam-A contains about 18,000 protein families. In 2018, Pfam-A residue coverage of the UniProtKB [2] was around 53% with about 20% of all UniProtKB sequences lacking any type of annotation. These numbers have been quite stable in the last five years, being almost the same in 2016 [42]. Similarly to structure-based databases, the portion of the sequence space covered by profile-HMMs obtained through human curation is limited by the amount of work that goes into building each family. As a consequence, even if databases such as Pfam-A can provide a broad description, they still cannot be fully exhaustive.

The Pfam-B database has been built using different algorithms, including ADDA (see Section 1.5.4.1). When choosing data to use as input for the automatic algorithms, any region already annotated in Pfam-A is removed, so that a double annotation is not possible. The last version of Pfam-B based on ADDA has been dismissed in 2015. In June 2020 Pfam announced on its blog (https://xfam.wordpress.com/2020/06/30/a-new-pfam-b-is-released/) a new Pfam-B version, with no clear details yet about the algorithm used to produce it, except that they used the protein clustering software named MMseqs2 [43], and that a publication on the topic is going to appear in the near future.

Pfam also collects families in larger groups, named *clans*. Families of the same clan must share a common evolutionary origin, mostly assessed using structural evidence or profile-profile information. Indeed, the similarity between two Pfam families can be scored using HHpred [44]. HHpred assesses the similarity between two profile-HMMs by evaluating the probability that the two profiles are representing the same family. This probability is expressed using a score that ranges from 0 to 100, where 100 means 1 in terms of true probabilities. The "relationship" table of a Pfam clan collects all its families in a graph, where edges are weighted according to the respective HHpred probabilities. We note that it is not uncommon to find clans where HHpred finds no similarity between profile-HMMs, and in particular where one or more profiles are completely isolated from the rest of the graph. These are usually cases in which structural information is used to determine clan membership of a family.

1.5.3 Interpro

Interpro [6] is a resource we specifically want to mention because of its nature as a meta-database. It is a database collecting information from several resources, including protein family databases such as Pfam or CATH-GENE3D, and many other annotations with different kinds of protein signatures. To do so, it uses models provided from these different databases. As an interesting feature, Interpro displays different annotations for the same protein sequences, showing differences in the annotation from databases built using different information.

1.5.4 Automatically-generated family databases and algorithms

An alternative approach to manually curated family classification is automatic, sequence-based clustering of protein regions. Unsupervised clustering generally results in a less accurate classification, but it has the advantage that high coverage of the protein sequence space can be achieved, at limited cost. Automated family classification has a long history in protein bioinformatics. In 1998 the ProDom [45] database was released, based on the MKDOM program [46], applied on about 20,000 proteins. The growth of protein sequence databases, together with more sophisticated bioinformatics methods and computational resources, has led over the years to the development of algorithms such as ADDA [7], EVEREST [8], and MCL [47], among others. The majority of these algorithms cluster full sequences rather than protein regions; ADDA and EVEREST are two of the few algorithms focused on protein regions, namely tackling the problem of defining protein families boundaries.

1.5.4.1 ADDA and Pfam-B

ADDA [7] is an algorithm first published in 2003 which searches families starting from an all-to-all set of BLAST alignments. As a first step, ADDA performs a sophisticated domain decomposition on each protein sequence, which aims to identify correctly the boundaries of each possible family appearing in the protein, especially for complicated architectures or in cases of sequence fragments. Subsequently it proceeds to cluster together such domains, building families.

Until 2015, the ADDA clustering algorithm was used to produce Pfam-B (see

Section 1.5.2), and it has since been discontinued due to the heavy computational cost of running ADDA on modern day, large sequence databases [42]. While for sure part of the bottleneck in ADDA is due to the all-to-all BLAST alignment generation, the domain decomposition procedure appears itself to be quite computational demanding.

A brief description of the ADDA algorithm As described in [7], the ADDA algorithm starts with an all-versus-all BLAST search using soft E-value thresholds (< 1). Alignments are represented in a graph, with sequences as vertices and alignments as edges. The graph is then cleaned by applying a 40 P.I. redundancy reduction over sequences (vertices).

Subsequently, domain cutting is performed by a two-step procedure. First, given a protein sequence, the domain boundaries are defined, so that intra-domain correlation is maximised and inter-domain correlation minimised. The algorithm selects as "splits" those residues in the sequence marking with a higher confidence the presence of two distinct domains. To estimate such confidence, ADDA uses the boundaries of the local pairwise alignments found in that sequence. Once defined the first possible split, further splits are found by iterating the procedure, obtaining as a result a set of nested putative domains organised in a tree. Secondly, domains are selected from the sets of putative domains, generated for each sequence. Such selection relies on an objective function modeling the block structure of BLAST multiple alignments; the objective function is a likelihood-function determining the likelihood of observing a specific pair of domains in two sequences sharing an alignment. Since the space of all possible domain partitions of all sequences is too large to enumerate exhaustively, ADDA uses a greedy optimisation strategy, through the pre-computed nested-domains trees. The optimization starts form the top of the tree and then iterates over the list of all domains in the tree. Descending one level in the tree, if the likelihood of the new partition increases, the split is accepted. This procedure is iterated, descending the trees until no additional cut in any domain increases the likelihood.

After the domains' boundaries are obtained, the sequence space graph is converted into a domain graph, where a vertex is a domain and an edge is an alignment between domains. Edges are weighted by the relative overlap between the alignments. If the alignment covers one of the domains by less than 20% of its length, the edge is removed; if a domain is linked to several adjacent domains, all found on another sequence, only those with highest overlap are saved, removing the rest of the edges. Such domain graph is decomposed into connected components, using a minimum spanning tree. Then, each domain in the same minimum spanning tree is considered as part of the same family.

1.5.4.2 EVEREST

EVEREST [8] has been proposed in 2006 and, similarly as ADDA, starts with an all-to-all BLAST search. The algorithm aims to collect together similar alignments to find homologous regions through transitive relationships. Also in this case a cautious refinement is performed, and in particular final families are selected using a machine learning strategy. Using as a training set the Pfam-A database, a general notion of protein domain is learned: this machine learning procedure is used to select, among all families found by the algorithm, the "good" ones. Therefore, EVEREST cannot be considered a purely unsupervised method. Similarly to ADDA, the procedure is sophisticated and, considering also the machine-learning component, it appears to be computationally demanding. Two databases have been published based on the EVEREST algorithm, in 2006 and in 2008.

A brief description of the EVEREST algorithm EVEREST uses as initial input a protein database (in Ref. [8] the Swiss-Prot database), on which an all-to-all BLAST search is performed, allowing for high E-Values. A first redundancy reduction is done using BLAST alignments: a protein is considered equivalent to another one (and neglected) if their BLAST alignment covers at least 95% of each, with E-value $< 10^{-90}$. Subsequently, EVEREST aims at removing repeats from proteins. Every protein is compared to itself, using an iterative variation of the Smith-Waterman algorithm, searching for repeated regions; all but the first and the last repeats found are removed, and the procedure is replicated to seek for other repeats in the protein. A list of possibly-similar pairs of proteins is then generated: this is obtained by asking for E-Values < 100 in BLAST alignments. Then, the above variant of the Smith-Waterman algorithm is applied to each pair of proteins in the list. Here, those segments that are significantly similar are collected into a segment database: therefore, each segment in the database has another segment paired with it. This database is a database of putative domains, with two similarity measures defined upon them: i) the sequence similarity between a segment and the one paired with it; ii) an overlap similarity between every two segments on the same protein (if any). Then, segments on each protein are clustered into groups according to the second measure, and the sequence similarity of the segments is inherited by their group, requiring every two segments that are in the same group to have an overlap higher than a certain threshold. This allows filtering against false transitivity induced by sequence similarity. These groups are clustered together according to their sequence similarity, using an average linkage algorithm [48]. At this point on each protein there are several segments belonging to a specific cluster. To consider the possibility of homo or multi-domain proteins, multiple occurrences of the same pattern are identified using the connected components in the graph built to perform average linkage clustering. Each connected component defines a domain in the family, and the boundaries of the domain are defined by the 40th percentile from outside of the boundaries of the segments in the connected component. After this cleaning procedure, each cluster of segments is a candidate family.

Finally, machine learning techniques are used to discard those families that are inappropriate. A randomly chosen set of half of the Pfam-A families is used as training set to infer the ”concept of family”, and then each cluster is scored according to the training set, removing the low scoring ones.

Chapter 2

The DPCfam protocol

DPCfam clusters together homologous regions of protein sequences obtained from pairwise alignments, using the Density Peak Clustering [9] at its core. This chapter is dedicated to the description of the DPCfam protocol and to present its implementations. In Section 2.1 we will describe the Density Peak Clusteing algorithm, at the basis of the protocol. In Section 2.2 we will describe the general workflow of DPCfam. Finally, in the last two Sections we will describe the two implementations of DPCfam we developed: one for small, proof-of-concepts clustering (Section 2.3) and one for large, all-to-all clustering of UniRef50 database (Section 2.4)

2.1 Density Peak Clustering

The Density Peak Clustering algorithm (**DPC**) has been proposed in 2014 by Rodriguez and Laio [9]. This algorithm is based on the idea that clusters correspond to density peaks, and that cluster centers are characterized by a higher density than their neighbours, together with a large distance from points with higher density. DPC is capable to cluster data in a non-parametric manner: it only requires that a mathematical distance can be defined on the dataset; moreover, it is in principle capable to detect clusters of any shape.

The first step of DPC consists in estimating, for each point i of the dataset, its local density ρ_i. The easiest way to do this[9] is to use a step-wise kernel estimator: namely, to count for every point i the number of its neighbours closer than a certain threshold distance μ:

$$\rho_i = \sum_j \chi(d_{i,j} - \mu) \tag{2.1}$$

where $\chi(a) = 1$ if $a < 0$ and zero otherwise.

In [9] the authors identify a simple and effective *rule of thumb* to decide which value of μ should give the best results on a given dataset: one can choose the threshold distance μ so that the average number of neighbors is around 1 to 2% of the total number of points in the dataset. Different datasets have different values for the μ to be used, but the rule is always the same.

The second step is finding, for each point i, the distance between i and its closest neighbour with higher density, here denoted as δ_i:

$$\delta_i = \min_{j:\rho_j > \rho_i} d_{i,j} \tag{2.2}$$

These two quantities, ρ_i and δ_i, are those that allow identifying density peaks. Indeed, a density peak is a local density maximum: this means that points that have no close neighbours with higher densities are reasonably density peaks; thus, any point with a high δ_i is a good candidate to be a density peak. On the other hand, a significant density peak should be dense, thus points with a small ρ_i should not be taken into account. This qualitative reasoning has been translated by Rodriguez and Laio into searching, by eyesight, for isolated points in the so-called Decision Graph, a scatter plot that puts into relation ρ_i and δ_i, respectively on the x and on the y axis.

An example of Decision graph in shown in figure 2.1: the top panel shows a toy-model dataset of 2-dimensional data, and the bottom panel its Decision Graph. As it can be seen, in the Decision Graph most of the data points are located close to the x axis, while only few stand out in the top-right section. These points, highlighted in red, are selected as density peaks, and are thus cluster centers. The decision about what point is reasonably an isolated point in [9] has been left to visual inspection.

The assignation of a point to a cluster is subsequently done by following variations in the values of ρ_i: each point is assigned to the same cluster as the one to which is assigned his closer neighbour with higher density.

The usage of DPC in DPCfam has to meet two requirements:

- to use the simpler, lighter and faster implementation of DPC: this is because DPCfam performs a clustering of a huge quantity of data and also repeats the clustering a huge number of times.
- to find a reasonable strategy to automatically select density peaks form the decision graph, in line with the previous point.

As an addendum, we needed to redefine the assignation procedure, not only to

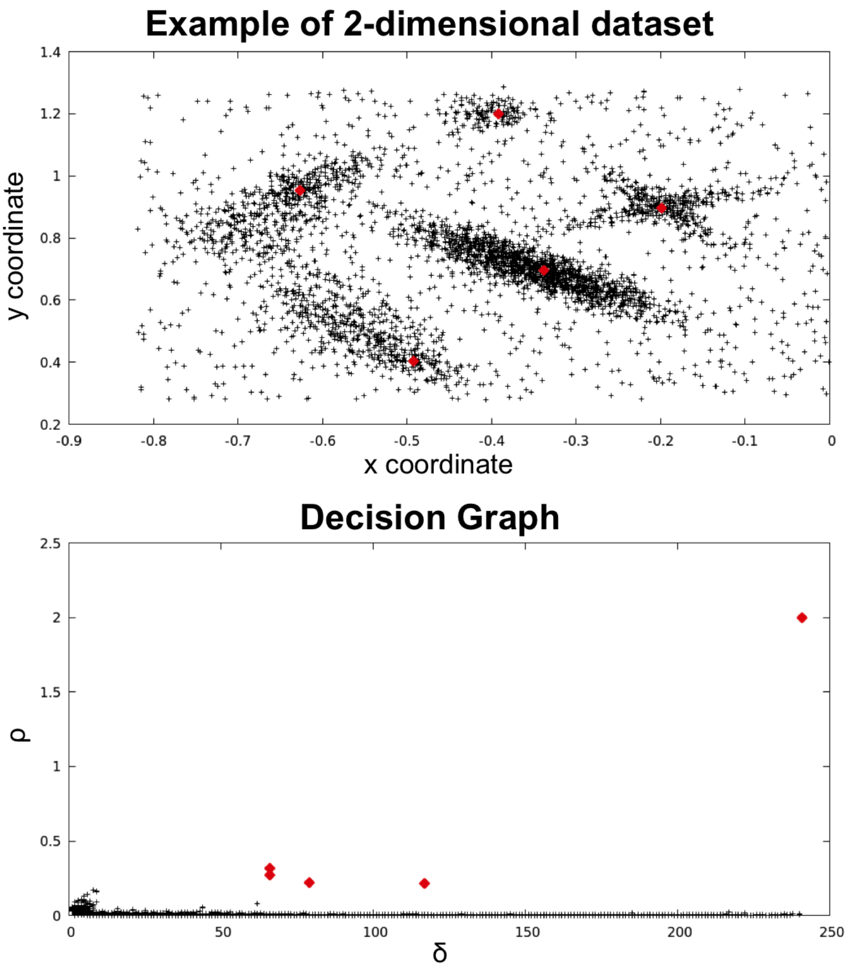


Figure 2.1: Example of a Decision Graph. Top panel shows a 2-dimensional distribution of points to be clustered using DPC. Bottom panel shows the respective Decision Graph, where the red points represent those selected as density peaks, highlighted in red also in the top panel to show their position in the dataset.

meet these two requisites but also to build clusters coherently with the purpose of automatically identifying protein families from alignments data.

Specific modifications of DPC used in DPCfam will be discussed in Section 2.2.2, when presenting DPCfam pipeline. In the next section we explain in detail the two strategies used to automatically select density peaks.

2.1.1 Automatic selection of density peaks: two strategies

In DPCfam we needed to automatically select density peaks from the Decision Graph. As we will explain better in Section 2.2, the DPCfam procedure is used on two different kind of data, subsequently. In the two cases, Decision Graphs shows notable differences, which suggested us to use different strategies to find isolated points.

2.1.1.1 The gap strategy

As done in [9], we introduce the quantity $\gamma_i = \delta_i \rho_i$. Being a density peak a point far from any other point with higher density (thus with large δ_i) and, being it a *density* peak (high ρ_i), points with high values of γ_i are good candidates to be density peaks. Surely, the point with largest γ_i will be a density peak; but what about other peaks? Up to which value of γ_i can a point be considered a density peak?

In the DPC paper the authors show that, looking at the sorted values of γs, the beginning of an anomalous growth in their values can signal the presence of density peaks. Starting from these considerations, we can select automatically density peaks by assuming that an "anomalous growth" corresponds to a gap in the sorted list $\Gamma = \{\gamma_s,\ \ \gamma_s > \gamma_{s+1}\ \ \forall s\}$ larger than a given threshold Δ. Using this strategy, we select as density peaks those data points whose γ is larger than a certain value γ_g, defined by the following condition:

$$\frac{\gamma_{g-1}}{\gamma_g} \geq 10^{\Delta}\ \ \&\ \frac{\gamma_{s-1}}{\gamma_s} < 10^{\Delta}\ \forall s > g\ \ \&\ g \leq g_{max} \tag{2.3}$$

Indeed, we search for the gap only between the first g_{max} elements of Γ, since we do not want to find an excessive number of clusters. Here Δ is a free parameter, which has been chosen as $\Delta = 0.5$. Note that in the bottom panel of Figure 2.1, isolated points highlighted in red have actually been selected using this automated procedure, using - as it will be done in the algorithm - $g_{max} = 20$. Varying Δ to smaller values results in more density peaks selected, while enlarging it will reduce them. For example, referring to the toy-dataset in Figure 2.1, using $\Delta = 0.5$ results in finding 5 peaks, while for $\Delta = 0.25$ we find 7 peaks. For $\Delta = 1.$ we still find 5 peaks and with $\Delta = 2$ we find only one peak. In general, as we will show, the results of the algorithm are stable if we vary Δ by 10% around the value of $\Delta = 0.5$, used in this thesis.

2.1.1.2 The maximum δ strategy

This method to find density peaks is strictly related to the kind of data to be clustered and to the distance $d_{i,j}$ defined on them: in particular, in this work we define two distances, both of them upper bounded to 1 ($\sup(d_{i,j}) = 1$). This maximum value has a well defined meaning: if the distance between two points is 1, then the objects that the two points represent have nothing in common, and can be considered as isolated from each other. Thus, we can state that any point having $\delta_i = 1$ is a point that is isolated from any other having larger density: it has, then, to be a local density maximum.

Such a consideration becomes very useful when a notable number of points in the dataset to cluster has $\delta_i = 1$: in this case, selecting any of these points as density peak is a fast and reasonable way to automatize the procedure. In the worst-case scenario, we might end up selecting very small peaks as clusters, that could be subsequently be trashed or merged to other clusters. To be consistent to the definition of a density peak, we also require the local density of such points to be larger than 1, being these points otherwise completely isolated from any other in the dataset, and thus meaningless.

2.2 DPCfam workflow

DPCfam aims to find a set of automatically generated protein families as a result of clustering together homologous protein regions obtained from local pairwise alignments. Such a task cannot be performed without a large and non-redundant reference database: in this work, we used the Uniref50 database [10], which contains proteins with less than 50 Percent Identity from UniRef, a database collecting virtually any known sequence from any known organism (see Section 1.4.1.1).

To find homologous regions between pair of proteins, we use BLAST (see Section 1.2.5), specifically the blastp program of the BLAST+ package [27]: however, other pairwise alignment software can be used. DPCfam performs a two-step clustering, using in both cases the DPC algorithm (see Section 2.1). The clustering procedure is blind to the nature and the quality of the pairwise alignments: everything it needs to know are the starting and ending positions of the alignments, both on the query and on the search sequences (see Section 1.2.2). This is the information needed to define a distance between alignments, firstly, and a distance between clusters of alignments, secondly. After the second run of DPC, the protocol applies a simple merging procedure aimed at eliminating redundancies, obtaining *metaclusters*.

Metaclusters are collections of homologous protein sequence regions: the final output of DPCfam is therefore a set of protein regions labeled with the respective metacluster. Such set can be both inspected directly or it can be used to build profile-HMMs (see Section 1.3), to be used to search in the protein sequence database and

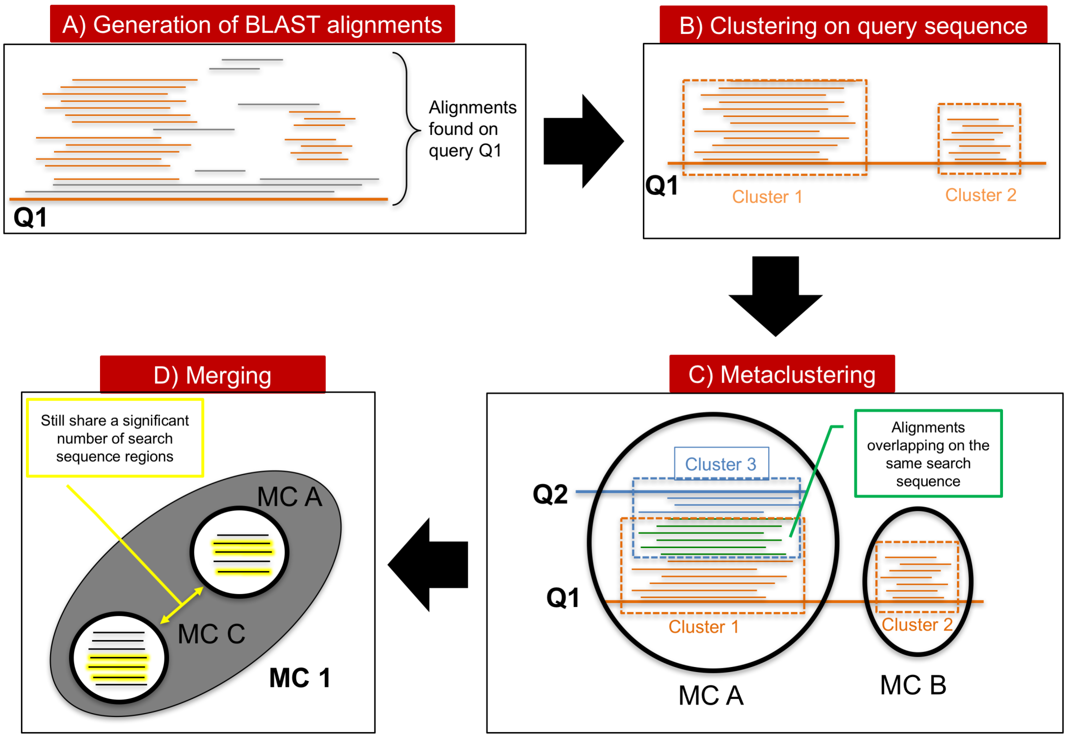

A) Generation of BLAST alignments
Alignments found on query Q1
Q1
B) Clustering on query sequence
Q1
Cluster 1
Cluster 2
D) Merging
Still share a significant number of search sequence regions
MC A
MC C
MC 1
C) Metaclustering
Alignments overlapping on the same search sequence
Cluster 3
Q2
Q1
Cluster 1
Cluster 2
MC A
MC B

Figure 2.2: Representation of the clustering and metaclustering process. Schematically, alignments that lie on the same region of the query sequence are clustered (B); subsequently, different clusters are "metaclustered", by considering close those clusters that contains a number of alignment overlapping on the same search sequence of the other cluster's alignments (C); finally, metaclusters are merged by grouping those that still share a significant number of search sequence regions (D).

collect all those sequences that have not been clustered directly by DPCfam, but are reasonably similar to the members of a metacluster.

A notable feature of DPCfam is its flexibility. While it can be used to find families in a large protein database (in this work, UniRef50), it can also be used to analyze subsets of query sequences containing several orders of magnitude of elements less than the full database, still being able to provide meaningful results, as we will show in Section 3.2. This is because the objects analyzed by DPCfam are pairwise alignments obtained by searching on the full database (UniRef50). These alignments include a vast number of search sequences that will add information not directly contained in the query set.

In the following subsections we will describe the DPCfam workflow, which consists of (see Figure 2.2):

- A) **finding BLAST** alignments of a query dataset on a protein database (Uniref50). As discussed above, the query database may be both the database itself or a smaller set of sequences, which in principle do not need to be a subset of the database.
- B) **primary clustering** of alignments lying on the same query sequence
- C) **metaclustering** of primary clusters
- D) **merging** of metaclusters to eliminate redundancies

Finally, we will explain the **filtering** procedure used to remove outlier sequences in a metacluster.

2.2.1 BLAST alignments

For each sequence (query) in the query dataset, we perform a local alignment search on the full UniRef50 database [10] using blastp from NCBI BLAST+ [27] (v. 2.2.30+). We save all alignments with E-value < 0.1 (see Section 1.2.3). We are interested in collecting a large number of alignments (when possible), and so we set the max_target_seqs option of blastp to 200,000 or 5,000,000, depending on the implementation.

We define here the *BLAST alignment* object, generally labeled by an index i, as:

$$B_i = \Big(q_i, s_i, \mathcal{Q}_i, \mathcal{S}_i\Big) \tag{2.4}$$

where q_i is the identifier of the query sequence, s_i is the identifier of the search sequence, $\mathcal{Q}_i$ and $\mathcal{S}_i$ are regions on, respectively, the query and the search sequence. More in detail, $\mathcal{Q}_i$ and $\mathcal{S}_i$ represent the boundaries (start and end positions) of the pairwise alignments on the query and on the search sequences, respectively (see Figure 2.3 A). Note that gaps and insertions are not taken into account.

2.2.2 Clustering of BLAST alignments

As explained in Section 2.1, DPC entails the following steps: i) defining a distance in the space of the objects that are to be clustered; ii) for each object estimating its local density; iii) selecting density peaks (cluster centers) and, finally, iv) assigning non-peak objects to density peaks (clustering). DPCfam performs two rounds of DPC: the first round (*primary clustering*) allows clustering alignments that cover

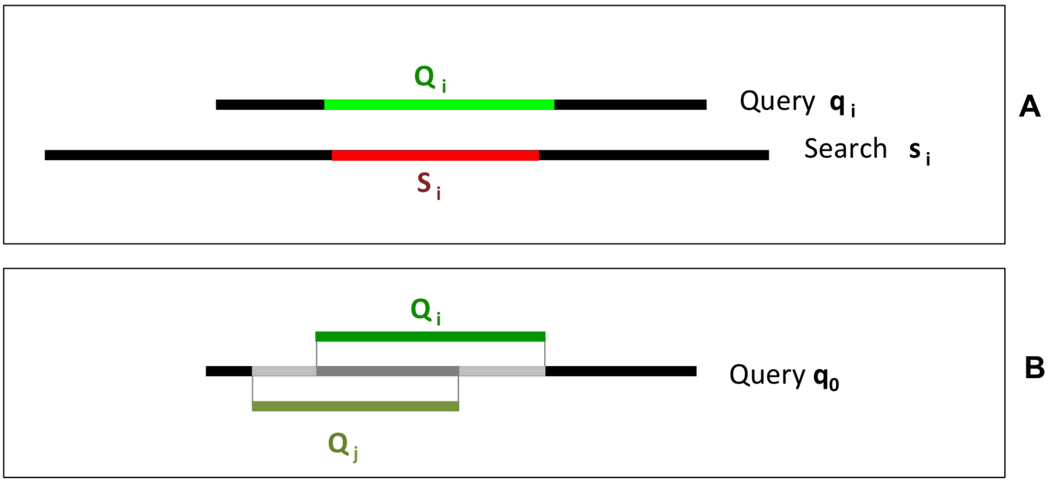


Figure 2.3: (A) Schematic representation of a pairwise alignment $B_i = (q_i, s_i, \mathcal{Q}_i, \mathcal{S}_i)$. The aligned regions are shown in green (query) and red (search). (B) Representation of two different alignments (B_i and B_j) on the same query q_0 . The aligned regions on the query are shown in green. The dark-gray portion of the protein represents the intersection between region $\mathcal{Q}_i$ and region $\mathcal{Q}_j$, namely $\mathcal{Q}_i \cap \mathcal{Q}_j$; the dark-gray+light grey region represents union of region $\mathcal{Q}_i$ and region $\mathcal{Q}_j$, namely $\mathcal{Q}_i \cup \mathcal{Q}_j$.

similar regions on the query sequences; the second round groups together primary clusters that share a number of overlapping alignments (*metaclustering*).

2.2.2.1 Primary clustering

For a query q_0 we write the set of all of its alignments as:

$$\mathcal{B}^{q_0} = \{B_i : q_i = q_0\} \tag{2.5}$$

We define the distance between alignments in $\mathcal{B}^{q_0}$ as:

$$d^{\mathcal{Q}}_{i,j} = 1 - \frac{|\mathcal{Q}_i \cap \mathcal{Q}_j|}{|\mathcal{Q}_i \cup \mathcal{Q}_j|} \tag{2.6}$$

where $|\mathcal{Q}_i \cap \mathcal{Q}_j|$ is the length (intended as number of residues) of the intersection between the segments identified by $\mathcal{Q}_i$ and $\mathcal{Q}_j$, while $|\mathcal{Q}_i \cup \mathcal{Q}_j|$ is the length of their union (see Figure 2.3 B). This distance is 0 if B_i and B_j are aligned to the same portion of the query q_0, that is, $\mathcal{Q}_i = \mathcal{Q}_j$; while it is 1 if $\mathcal{Q}_i$ and $\mathcal{Q}_j$ do not overlap at all. As defined, $d^{\mathcal{Q}}_{i,j}$ represents a metric since it is symmetric, positive defined and satisfies the triangular inequality.

Using the distance in Eq. 2.6, we estimate the density ρ_i of the alignment i, using the estimator in 2.1:

$$\rho_i = \sum_j \chi(d_{i,j} - \mu_1) \tag{2.7}$$

Thus, the density of an alignment B_i is given by the number of alignments that belong to the same set $\mathcal{B}^{q_0}$ and that are found at a distance from B_i smaller than μ_1. In the algorithm we set $\mu_1 = 0.2$, according to the rule of thumb discussed in Section 2.1. When two alignments with the same search sequence are such that $d^{\mathcal{Q}}_{i,j} < \mu_1$, we retain only the alignment with the lowest E-value. For each query, this happens in 1% or less of the alignments.

Next, we search for density peaks: in this step, we use the gap strategy (see Section 2.1.1.1): we sort the alignments according to decreasing values of γ_i, $\Gamma(q_0) = \{\gamma_s,\ \ \gamma_s > \gamma_{s+1}\ \forall s\}$, and we select density peaks by identifying a $\gamma_g \in \Gamma(q_0)$ such that $\frac{\gamma_{g-1}}{\gamma_g} \geq 10^{\Delta}\ \ \&\ \frac{\gamma_{s-1}}{\gamma_s} < 10^{\Delta}\ \forall s > g\ \ \&\ g \leq g_{max}$. We recall that the number of peaks we allow is limited by the parameter g_{max}, which is set to 20. Such a decision is derived from the belief that it is very improbable for a protein to contain such a large number of distinct families. A peculiar case is the one of repeats: sometimes a protein can contain tens of repeats, which means that the g_{max} we chosen will not allow to separately identify all of them as primary clusters. However, the fact that we cannot identify by design every single occurrence of a repeat in a query protein is not problematic: through the metaclustering we will still be able to get a description of a repeated family, thanks to the vast number of alignments collected together. In return, using a relatively small value of g_{max} allows us to limit as much as possible the noise which may occur by using instead larger and more permissive values.

Once we detected the density peaks, we proceed to assign the non-peak data to a cluster. To do this, we assign to each density peak all alignments that are found at a distance smaller than μ_1 from the peak, and further away from any other peak: the set of alignments assigned to a peak constitutes what we call a *primary cluster*. This procedure differs from the standard DPC assignation procedure [9], because not all alignments are assigned to a primary cluster, but only those close to a peak. Indeed, we aim to collect together sequences that are close each other, which helps us to identify an homology region on the protein, where the clustered alignments will lie. We then discard the non-clustered alignments from the rest of the analysis. Thus, the clusters we obtain are subsets of the previously defined $\mathcal{B}^{q_0}$ set, where each subset includes alignments located around the same region of the query sequence.

The clustering procedure we described is schematically shown in Figure 2.2 (A and B), and two examples of primary clustering are shown in Figure 2.4.

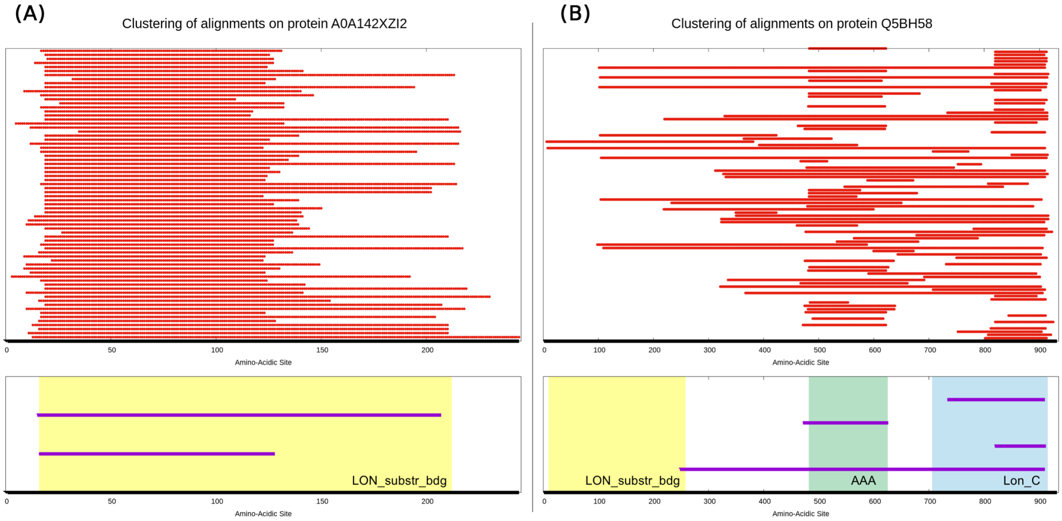


Figure 2.4: Examples of primary clustering results for two proteins from namely A0A142XZI2 (A) and Q5BH58 (B). Thick, black lines represent the query sequence. Red lines in the top section of each panel show a random subset of the regions of the query that have been aligned by BLAST to other sequences in the search set. The bottom part of each panel shows instead a comparison between the protein's Pfam annotation and the primary clusters obtained form the query sequences. Clusters' centers are represented as purple lines and vertically ordered with respect to their ρ value.

2.2.2.2 Metaclustering

The goal of this step is to collect together primary clusters that share a large number of regions (specifically, regions of search sequences) together. What we will do is to perform once again a Density Peak Clustering, clustering not the alignments but the set of primary clusters we found on the query set. We will, then, need to define a distance between primary clusters.

We denote the set of alignments belonging to a primary cluster c as $\mathcal{B}_c$ and we call N_c the number of its elements.

We then define the distance between two primary clusters c and c_0, associated to two queries q and q_0 as:

$$D_{c,c_0} = 1 - \frac{1}{min(N_c, N_{c_0})} \sum_{m\in\mathcal{B}_c, n\in\mathcal{B}_{c_0}} \delta_{s_m s_n} \chi_{\mu_d}(d^{\mathcal{S}}_{m,n}) \qquad (2.8)$$

where $d^{\mathcal{S}}_{i,j}$ is defined as in Eq. 2.6 using segments $\mathcal{S}_i$ and $\mathcal{S}_j$ in place of $\mathcal{Q}_i$ and $\mathcal{Q}_j$, and $\mu_d = 0.2$ is chosen coherently with μ_1 in Eq.2.6. This distance is shorter the higher the number of alignments in the two clusters sharing the same search sequence and having a significant overlap, namely $d^{\mathcal{S}}_{i,j} < \mu_d$. Also in this case, we have $D_{c,c_0} = 1$ when the two primary cluster have nothing in common; if $D_{c,c_0} = 0$ the smaller of the two clusters is contained in the larger.

We estimate the density ρ_c, again using the estimator described in Eq. 2.1:

$$\rho_c = \sum_{c'} \chi_{\mu_2}(D_{c,c_0}) \tag{2.9}$$

where $\mu_2 = 0.9$ was again chosen following the rule of thumb in [9].

We then proceed computing

$$\delta_c = \min_{c':\rho_{c'}>\rho_c} D_{c,c'} \tag{2.10}$$

and selecting the density peaks. In this step, we use the maximum δ strategy to select cluster centers, which consists in choosing as density peaks all points having $\delta_c = 1$ and $\rho_c > 1$ (see Section 2.1.1.2). Now we can explain why we prefer this strategy to the gap strategy, used in primary clustering, by focusing on the different goals of the two procedures. Primary clustering aims to identify, on a given query sequence, regions and respective alignments that could be reasonably be part of a "family", which to us correspond to those regions where we found several alignments. Metaclustering clusters together these sets of alignments, with the purpose of building proper families. Protein families should be as separated as possible, which in our algorithm can be easily achieved asking for cluster centers (density peaks) to have $\delta_c = 1$ (see Section 2.1.1.2). As we will see, this greedy strategy still leads to a too large number of metaclusters, and we need a merging step to reduce this redundancy. This excludes the necessity to use more sophisticated strategies such as the gap strategy to select density peaks.

Again, we assign to each density peak all primary clusters that are found at a distance smaller than μ_2 from the peak, and further away from any other peak.

The set of primary clusters assigned to a peak constitutes a raw *metacluster*. Primary clusters not assigned to any peak are discarded.

2.2.2.3 Merging metaclusters

DPC metaclustering is very conservative in assuring that primary clusters collected together are similar each other; however, different raw MCs can contain primary clusters very similar each other, which are considered redundant. We mitigate this redundancy by adding, after metaclustering, an extra merging step. We merge similar raw MCs by computing the quantity

$$D_{MC',MC''} = \frac{2}{N_{MC'}N_{MC''}} \sum_{c' \in MC', c'' \in MC''} D_{c',c''} \tag{2.11}$$

where MC' and MC'' are any two metaclusters and $N_{MC'}$ and $N_{MC''}$ is the number of their primary clusters. $D_{MC',MC''}$ is the average of the distances between primary clusters contained in the two MCs, where $D_{c',c''}$ is computed following Equation 2.8. We merge all MC pairs for which $D_{MC',MC''} < 0.9$.

2.2.2.4 Filtering metaclusters' alignments and building profile-HMMs

A metacluster is a collection of protein regions $\mathcal{S}_i$. In order to reduce the level of noise coming from outlier sequences within a MC, from the list of all regions obtained in the previous section we remove those that don't overlap with any other sequence in the MC. More specifically, we keep region i if it exists another region j in the same MC such that $\delta_{s_i s_j} \chi_{\mu_d}(d^{\mathcal{S}}_{i,j}) = 1$ (cfr. Eq. 2.9). We additionally reduce redundancy at 95 percent identity using Cd-Hit (v4.7) [37].

For the purpose of building MC-associated profile-HMMs, we further reduce MCs' size by reducing redundancy at 60% using Cd-Hit.

2.3 DPCfam0: a Python/C++ implementation for proof of concepts analysis

DPCfam0 constitutes the first implementation we wrote of the DPCfam procedure. DPCfam0 is capable to build metaclusters starting from a relatively small set of pairwise alignments. In particular, we used it to cluster the PUA_UR50 and the P53_UR50 datasets (see Section 3.2) This software repository is published at

https://gitlab.com/ETRu/dpcfam0

2.3.1 General description of the implementation

DPCfam0 is constituted by a Python code orchestrator (*DPCF.py*) and a set of C++ programs performing the most computationally expensive tasks of the protocol. In addition, it includes the possibility to run a BLAST search (provided a blastp program is installed) to generate pairwise alignments of a query set with respect to a protein database. Each step of the protocol is separated and can be ran independently, provided the input files needed exists and are located in the correct folder.

The three C++ programs are devoted to:

- perform primary clustering (*primarycl.cc*) of BLAST alignments
- build the distance matrix among primary clusters (*link.cpp*)
- perform secondary clustering on the distance matrix (*secondarycl.cc*)

These programs are not encapsulated in the Python code: we compile them separately, and then we use the *subprocess* library in Python, which directly calls the programs each time through a shell command. In a similar manner, the blastp command is called (if needed) by the Python orchestrator. Beyond this, the Python ochestrator manages the input and output file, performing files parsing and transformations when needed, and finally it performs the filtering of the MCs produced by the secondary clustering C++ program.

In the following we will give a brief description of each step of this implementation.

2.3.2 Generating BLAST alignments

DPCfam0 uses blastp to generate local pairwise alignments to be clustered. To do so, a protein sequence database is needed (in our case, UniRef50) using numerical (integer) IDs for fastening the process. We collect all the query sequences of interest in a fasta file, and we search using blastp setting the E-Value threshold to 0.1 (see Section 1.2.3) and the max_target_seqs parameter to 5,000,000. This last parameter allows us to extract virtually all the local pairwise alignments possible: indeed, max_target_seqs indicates the maximum number of alignments per query, and its default value is 500. In fact, when searching for alignments one is usually interested in a reduced number of results - but in our case, we want to extract as much as possible relevant local pairwise alignments. To do so, we enlarge the maximum value of this parameter by several orders of magnitude. This request can slow down the BLAST search, but we gain a large quantity of alignments to work with.

The output produced by blastp is a single tab-separated file, each line representing an alignment. For each alignment, we save the following data: the query sequence ID, the search sequence ID, start and end position of the query aligned region, start and end position of the search aligned region; moreover, other data about the alignment are outputted, such as the query's and search's lengths, the alignment's length, the E-Value, etc. Finally, we also get both the query aligned sequence and the search aligned sequence.

2.3.3 Primary clustering

The primary clustering program (*primarycl.cc*) works on a single query sequence. Therefore, the Python orchestrator first splits the blastp output in a series of files, each containing alignments derived from one specific query. Then, the C++ program is called by the orchestrator for each of these files.

primarycl.cc computes the matrix distance between the query's alignments (see Eq. 2.6) and performs the density peak clustering using the "gap strategy" to find cluster centers (see Section 2.2.2.1). First of all, a single query alignments file is read. The algorithm computes the distance $d^{Q}_{i,j}$, and also the quantities needed for clustering, ρ_i (see Equations 2.1 and 2.7) and δ_i (see Equation 2.2.). A first cleaning step removes those alignments which, obtained form the same search sequence, overlap too much each other: only those with the best E-value is saved. Then, the local density is estimated for each alignment: first, it is set to a value around to 1 (specifically, the score of the alignment divided by its length, in order to give a small priority to those alignments with better score); then, for each couple of alignments i, j we compute the distance $d^{Q}_{i,j}$ and we add 1 to ρ_i and ρ_j if their distance is smaller than the distance kernel μ_1. We add to the distance computed an extremely small (OOM 10^{-5}) pseudo-random noise from a uniform distribution to avoid degeneracy, namely cases in which $\delta_i = \delta_j$ and $\rho_i = \rho_j$. We then compute the δ_i of the alignments, searching for each alignment i those alignments with ρ_j larger than ρ_i , and then selecting the smaller distance between such i and j. Here again the distance is re-computed, adding the small noise to remove degeneracy. We note that noise is added using the RAND() C++ function, initialized with an hard-coded seed at the very beginning of the program. Thus, the results are deterministic with a specific input-file.

Once ρ_i and δ_i are computed, the program searches for the cluster centers (or density peaks) following the "gap strategy" (see Section 2.1.1.1). We select the first 20 alignments with higher $\gamma_i = \rho_i \delta_i$, sorted according to this quantity, as putative centers. For each consecutive couple we compute the gap and we check if it is larger (as a logarithm) than 0.5; we track the last one with this property and we remove from the center's putative list all those centers listed after the last relevant gap. Those remaining are the cluster's centers.

To assign an alignment to a cluster, we compute the distance between the alignment and each cluster center: as we go through the centers' list, we save the match with the smaller distance. At the end of this search, we print in the output the alignment assigned to the closer cluster center, provided that the distance is smaller enough ($< \mu_1$, see 2.2.2.1); otherwise, the alignment is not reported in the output.

As a final output is produced a file for each query. Such file contains in each line information about an alignment that has been clustered for that query.

Using the Python orchestratorl, we split each output file in as much files as the number of clusters found in the respective query: we introduce here a labeling system to univocally define primary clusters. Therefore, a primary cluster is labeled using the following number:

$$c_{id} = qid * 1000000 + cen_{id} \tag{2.12}$$

where qid is the query ID (numeric) and cen_{id} is a numeric id assigned to the cluster's center alignment.

2.3.4 Secondary Clustering

In DPCfam0 the "secondary clustering" includes both the metaclustering and the metaclusters' merging steps (see Sections 2.2.2.2 and 2.2.2.3). In detail, we separated the procedure in three sections:

- the generation of the distance matrix between primary clusters (*link.cpp*)
- actual metaclustering (*secondarycl.cc*)
- merging (done by the Python orchestrator)

2.3.4.1 Distance matrix generation

To better understand the implementation done in *link.cpp*, it is useful to consider the meaning of Equation 2.8, defining D_{c,c_0}. To compute this distance, we basically need to compare the alignments contained in two primary clusters (c and c_0), and check for any couple of alignments (each from a different cluster) deriving form the same search sequences ($\delta s_m s_n$). Given this match, we check if the two alignments are close enough ($\chi\mu_d$, with $\mu_d = \mu_1 = 0.2$): if this is the case, we add 1 to a "total" which will then be normalised to the maximum possible number of matches. The quantity obtained so far, $\tilde{D}$, allows computing the distance as $D_{c,c_0} = 1 - \tilde{D}_{c,c_0}$. Moreover, we note that if there is an alignment belonging to primary cluster c having search protein s_m and there is no other alignment in cluster c_0 with same search sequence s_m, this alignment will not contribute in the computation of D_{c,c_0}. Indeed, any contribution to the distance will be obtained only from couples of alignments sharing the same search sequence.

To implement this computation in an efficient manner, we preprocess the primary cluster data (via the Python orchestrator) as it follows: we collect in a single file all the clustered alignments, assigning each the primary cluster label defined in Equation 2.12. Then we sort the alignments with respect to their search sequence numerical identifier. From such a list, we can easily discriminate those alignments whose search sequence appears only once in the full list: as said before, such alignments won't contribute to the computation of any entry in the distance matrix D_{c,c_0}, and we can discard them. The rest of the alignments we collect (and sort) constitute the input to the *link.cpp* C++ program. To compute the distance, the program collects in a vector a consecutive set of alignments sharing the same search ID. Such groups of alignments are relatively small, and an all-to-all search within them is much faster than an all-to-all performed on the full set of alignments. Thus, given a group of alignments sharing the same search ID, for each couple m, n we compute the distance $d^{\mathcal{S}}_{m,n}$, and, if it smaller than μ_d, we add 1 to the matrix element associated to the alignments' primary clusters. In this way, we are computing $\tilde{D}$ (yet to be normalized). Finished this all-to-all search the vector we used is cleaned, and another group of alignments with the same search ID is loaded from the input file. At the end of the alignment's input file, we know that there is no other contribution to the distance matrix to take into account, and so we proceed to normalize $\tilde{D}$ and compute the distance as $D = 1 - \tilde{D}$. The final output produced by this step (after some manipulation done by the Python orchestrator) is a triangular matrix in the form i,j,$D_{i,j}$. i and j are novel IDs for clusters, from 1 to C (being the latter the number of primary clusters). A mapping from this IDs to those defined in equation 2.12 is saved in a separated file, containing also the size (N_c) of the primary clusters.

We note that the main problem of *link.cpp* (solved in the parallel optimization, described in Section 2.4) is that we allocate the full distance matrix in memory.

2.3.4.2 Metaclustering

Metaclustering of primary clusters is performed directly by *secondarycl.cc* program, which uses the previous triangular matrix (i,j,$D_{i,j}$ form) as an input: again, the entire matrix is loaded in memory. The program is then very similar to *primarycl.cc*, computing ρ and δ. Finally, cluster centers are selected using the maximum δ strategy. The rest of the points are assignated to the closest density peaks, provided that distance with it is smaller than $\mu_2 = 0.9$ (see Section 2.2.2.2). Two output files are printed: one contains the number of primary clusters in each MC, the other reports to which MC each primary cluster has been assigned.

2.3.4.3 Merging and Filtering

The final merging and filtering step is performed by the Python orchestrator. Using files produced so far, it first computes the MC-MC distances (see Equation 2.11). Then, for each MC, it searches for other MC closer to it than the merging threshold (0.9), and merges them iteratively.

A new version of the previous output file is written, containing the information of the new, merged, MCs.

We then proceed to write the single MCs' files, filtered. Filtering procedure consists in printing only those sequences that had generated a contribution to the distance between primary clusters of the same MC. To do so, we apply a strategy very similar to the one we used to compute the distance matrix D_{c,c_0}: we collect all the alignments of a metacluster and we sort them with respect to their search ID. Search IDs appearing only in one alignment tells us that such alignment couldn't contribute to the computation of the distance, and then we remove them. Then, we select groups of alignments with the same search sequence, and for each of them we check if there exists at least another element in the group with $d^{\mathcal{S}}_{m,n} < \mu_d$: if this is the case, we save that alignment. Once we have the list of all the "good" alignments in the MC, we trace back to their proteins' sequences. For each MC, we output a file containing the aligned region of the search sequence of each "good" alignment. Each sequence is labeled using the search ID and the start-end position on it of the aligned sequence reported.

2.4 Parallel and optimised DPCfam implementation

In the following we will discuss some of the main aspects of the DPCfam implementation that has been used to cluster protein sequences in the UniRef50 database (v. 2017_07), which contains 23,531,980 proteins. Here UniRef50 serves both as query set and search database.

The complete run of this DPCfam implementation, applied to pairwise alignments, costed about 40,000 CPU hours. The generation of such pairwise alignments, namely the blastp all-to-all search on UniRef50, costed about 190,000 CPU hours.

Results obtained from this implementation are discussed in Section 3.3.

2.4.1 Generating BLAST alignments

As a very first step, we shuffle the UniRef50 database and assign to each sequence a numeric, integer and random ID, from 1 to 23,531,980. In particular we ”sort” the list of the proteins’ names using the random (”-R”) option of the *sort* bash program; we assign to each sequence the numerical ID corresponding to the row number of its protein name after shuffling. This step is necessary to balance the computations to be done in the following; moreover, using numerical IDs both fastens the BLAST search and helps the optimization of the DPCfam protocol.

We generate local pairwise alignments using the *blastp* program of BLAST+ (v. 2.2.30) suite; we run it in parallel by using the program’s native multithreading system together with a simple bash command parallelization. We set the maximum E-value allowed for the alignments to 0.1 and the max_target_seqs option to 200,000: this last choice limits to 200,000 the number of results obtained when searching a query against the UniRef50 database. We note that the value we use (200,000) is higher than the default value offered by BLAST (500) but lower than the value used in Section 2.3.2 (5,000,000). We opted for this middle ground solution in order to speed-up the computation: limiting the per-query results to 200,000 matches is a reasonable compromise between generating a useful number of pairwise alignments and limiting the time required by this step.

Alignments have been obtained using the local cluster, on Xeon E5-2680 v2 nodes (2 sockets, 10 cores, 2 threads, 40GB of RAM).

We run blastp using about 100,000 query files, containing 200 query sequences each. Each query file is searched against the (shuffled) UniRef50 database, prepared with the *makeblastdb* program of the BLAST+ suite. We set the blastp multithreading system to 4 threads (-num_threads 4) and run in parallel 4 blastp commands for each node, resulting in 16 nodes fully occupied by the blastp program. To speedup the I/O, we load on the local node a entire copy of the UniRef50 database.

As an output we report mostly data useful to the DPCfam protocol on an ASCII tab-separated file. Specifically, we save 12 columns: the query sequence ID, the search sequence ID, start and ending positions of the alignment on the query sequence, start and ending positions of the alignment on the search sequence, the lengths of the region aligned on the query sequence, the lengths of the region aligned on the search sequence, the alignment length, the P.I. of the alignment, the E-value and the score.

Each file contains the alignments obtained from 200 query sequences. As a result, we obtain about 3.5 TB of data, containing about 55 billions of local pairwise alignments.

2.4.2 Primary clustering

The primary clustering has an embarrasingly parallel workload. For each query sequence we need to analyse only the subset of alignments obtained from that specific query. Such a subset is a partition of the whole set of alignments we previously produced, which makes the primary clustering performed for query q_a totally independent from the one performed for query q_b. Primary clustering is implemented as a C++ code that takes, as input, the blastp output as described before; it is run in parallel using MPI [49]. MPI usage is limited to pass to each process a number of query sequences (and relative alignments) to cluster, sequentially. This step took around 2,500 cpu hours, obtaining about 33 billions of alignments clustered in 27 millions of primary clusters. At this level, reducing as much as possible the files' size becomes essential: on one hand, because of disks capacity, on the other, because time spent loading data from hard-disks to RAM became relevant in the subsequent computations. The output produced by primary clustering has been reduced to the essential data needed for the rest of the clustering procedure: for each clustered alignment we save the query ID, the search ID, a cluster identifier and finally its start and end positions on the search sequence. To reduce further the files' size, we convert them to a binary representation intended to use a minimal number of bits for each column. For example, proteins IDs takes integer values that cannot be larger than 23,531,980 (number of total proteins in UniRef50): therefore search IDs can be represented using a particular C++ variable type, "uint32_t", which "costs" only 4 bytes and where such numbers easily fit. Similarly, start and end positions of the alignments are integer values that ranges from 0 to about 30,000, and can be represented using the "uint16_t" variable type, which costs 2 bytes. To label clusters, we use a similar method as in Equation 2.12, with some precautions. Since for each query the maximum number of clusters is 20, by using numbers from 1 to 20 to label a query's cluster c_{id}^{query} we can use as a generic cluster label $c_{id} = qid * 100 + c_{id}^{query}$. In this way, c_{id} is a positive integer smaller than 2,353,198,020, a number that can still be represented using the "uint32_t" variable type. With this strategy, we finally can represent clustered alignments using 16 bytes for each alignment, using about 500GB to store all of them.

2.4.3 Generating the distance matrix between primary clusters

The distance matrix between primary clusters is needed for the subsequent metaclustering: generating it is computationally heavy and also RAM occupation during the computation became an issue. The reasoning done in Section 2.3.4.1 is useful also in this context: we report it also here for sake of readability. To compute the distance D_{c,c_0} (see Equation 2.8), we need to compare alignments contained in two primary clusters (c and c_0), and search for any couple of alignments (each from a different cluster) having the same search ID ($\delta s_m s_n$). Given this match, we still need to check if the two alignments are close enough ($\chi \mu_d$, with $\mu_d = \mu_1 = 0.2$): if this is the case, we add 1 to a "total" which will then be normalised to the maximum possible number of matches. The quantity obtained so far, $\tilde{D}$, allows computing the distance as $D_{c,c_0} = 1 - \tilde{D}_{c,c_0}$. In DPCfam0 we worked with clustered alignments sorted with respect to the search ID (see Section 2.3.4.1): this allowed us to collect sets of alignments with the same search ID. For each couple of alignments in these sets, we added their contribution to the respective primary clusters' $\tilde{D}$ element (if the aligned regions were close enough). In the all-to-all scenario we deal both with a massive input and a massive output: we have 500GB of input data containing alignments grouped in $27 \cdot 10^6$ primary clusters, leading to a distance matrix of $(27 \cdot 10^6)^2$ elements (sparse). Moreover, sorting all alignments with respect to the search ID is a heavy and expensive task. Using the exact strategy as used in the DPCfam0 implementation is not feasible.

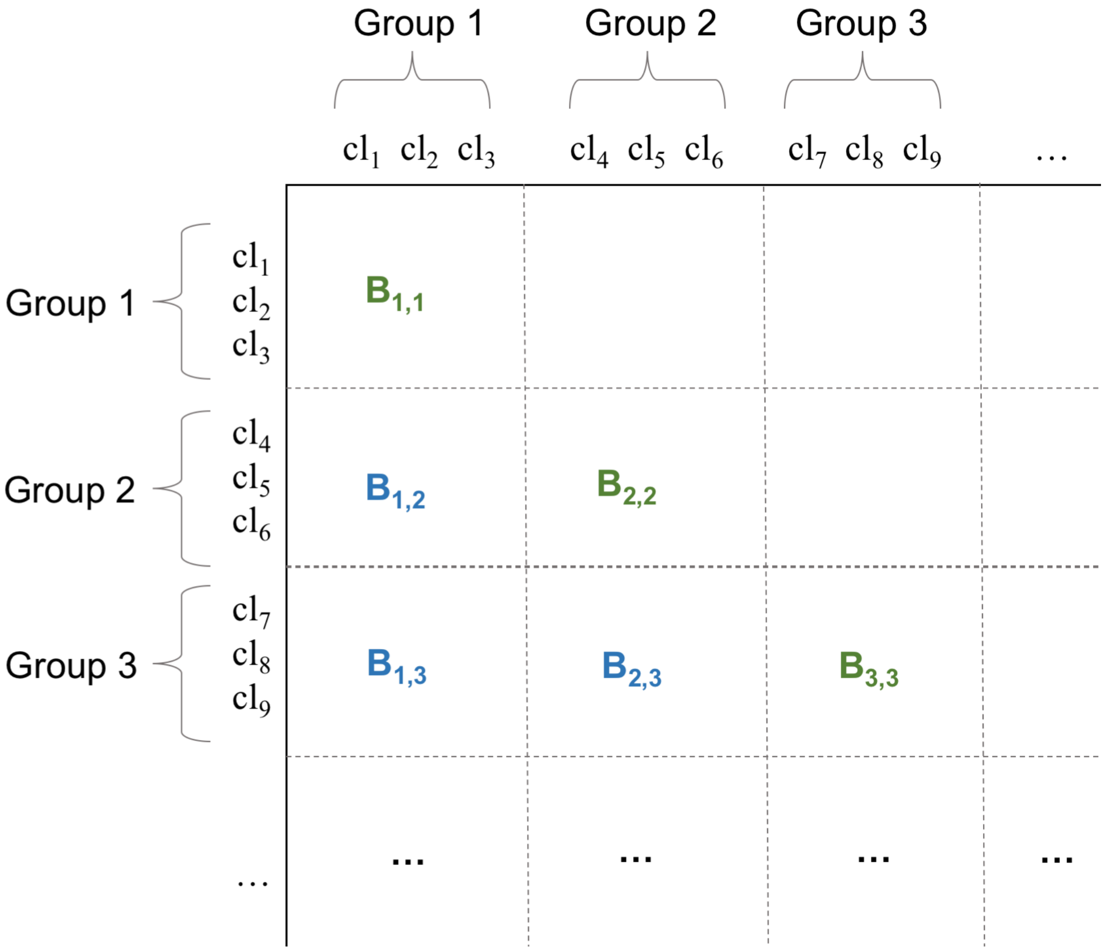


Figure 2.5: Schematic representation of distance matrix divided in blocks. Here we have groups of (about) 3 primary clusters, and to each couple of groups corresponds a block (included couples made of the same group). Since the matrix is symmetric, we can consider a single block for each couple. Blue blocks represent off-diagonal blocks, green blocks represent diagonal blocks.

We decided to split the problem in a set of sub-problems, which also allows us to proceed in parallel. Let us first consider the distance matrix we are going to build: as shown in Figure 2.5 we can split it in blocks by dividing primary clusters in N groups, obtaining N^2 blocks. The content of each block can be computed independently, and we need to actually work on $\frac{N(N+1)}{2}$ blocks, being the distance matrix symmetric.

The computing strategy is still similar to the one adopted in DPCfam0, with the difference that instead of using a single input file we will use two input files for each block. First, we divide primary clusters in N groups. This is an easy task: indeed, primary clustering output is already organized in groups of queries. Since primary clusters are defined on a query, each group of queries will contain a group of primary clusters. Moreover, we can expect to find in each group of queries more or less the same number of primary clusters, obtaining balanced groups of primary clusters from the previous output. Within each group we sort the alignments with respect to the search ID: this is a much easier tasks than sorting all 33 billions alignments together. Here use a C++ program to sort the binary files, based on the radix sort algorithm [50]. We exploit this sorting to search for sets of alignments with the same search ID, so that we can compute their contribution to the $\tilde{D}$ elements of the primary clusters. Then, to compute the matrix block $B_{i,j}$, we read simultaneously the two files containing alignments of group i and of group j. Our goal is to identify a set of alignments in group i and another in group j that share the same search ID. The first alignments found in each file will have search ID $s_{i,1}$ and $s_{j,1}$, respectively. Two cases are possible:

- $s_{i,1} = s_{j,1}$: we found a match between search IDs in the two files. We continue reading the files collecting alignments in two vectors, v_i and v_j, stopping to collect them when the search IDs change. The two vectors may contain a different number of elements. For each alignment in v_i we compute the distance $d^{\mathcal{S}}$ with any other alignment in v_j: if the distance of two alignments is smaller than 0.2, we add one to the $\tilde{D}$ entry of the respective primary clusters.

- $s_{i,1} > s_{j,1}$ or $s_{i,1} < s_{j,1}$: this means that the search IDs we found are different, and necessarily one is larger than the other. Let us consider the specific case where $s_{i,1} > s_{j,1}$: since alignments are sorted in both files, this means that in group i there is no alignment with a search ID smaller than $s_{j,1}$. Therefore, we can scroll the lines of group i file until we find $s_{i,n} \geq s_{j,1}$. Specifically, if $s_{i,n} = s_{j,1}$, we can compute distances (see previous point); if $s_{i,n} > s_{j,1}$ we find ourselves in the same situation as before, requiring this time to scroll trough the lines of group j file searching for $s_{j,m} \geq s_{i,n}$.

The reading procedure is schematically represented in Figure 2.6. At the end of the two input files, we normalize the values of $\tilde{D}$ and we convert them in entries of D, recalling that $D = 1 - \tilde{D}$.

We can distinguish between two types of blocks (see Figure 2.5): off-diagonal (blue) and diagonal (green). When computing diagonal blocks, we change slightly the procedure: we add an "if" condition checking that we are not working on couples

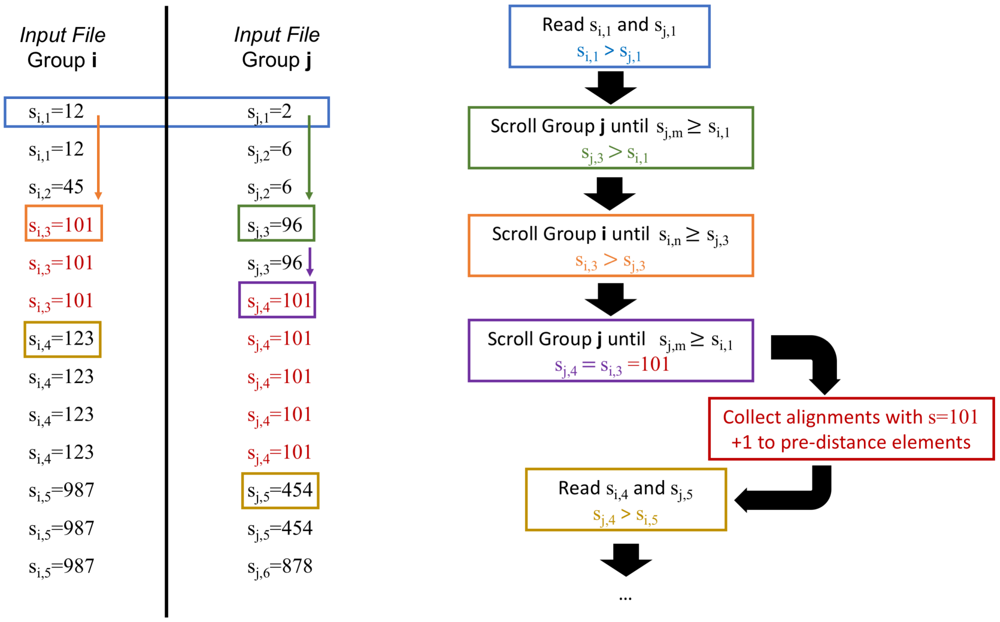


Figure 2.6: Schematic representation of the strategy used to compute a block of the distance matrix. We highlight the reading procedure that allows to find sets of sequences sharing the same search ID, which are those that will contribute to compute the matrix elements.

of alignments from the same primary cluster. This allow to reduce the number of comparison to be done between diagonal blocks.

We note that it is very important to set a balanced number of groups N. Indeed, if N is small ($N = 2$ as extreme case) we will need to store in memory the entire distance matrix in the process. This, we estimated, would require several Terabytes of RAM. In this case, the gain is that we need to compute only three blocks: $B_{1,2}$, $B_{1,1}$ and $B_{2,2}$. On the other hand, with a large N we need less RAM for each computation, but we need to compute $N(N+1)/2$ blocks. While having a large number of independent blocks is useful to compute them in parallel, we must consider also that with more blocks we are increasing the I/O required by the full computation. Consider the case of $N = 2$: each group of alignments will weight 250 GB, and we will compute 3 blocks loading two groups each time. At the and of the computation we'll have moved from disk to RAM 1TB. If we double to $N = 4$, our groups will weight 125GB each, but we will compute 10 blocks, moving from disk to RAM 2.5 TB. If N becomes too large, the time took by I/O becomes dominant and slows down the entire process. There is also another reason why we prefer a smaller number of groups: the strategy we use to compute the matrix elements relies on finding sets of alignments sharing the same search ID. As the number of groups increases the number of sequences they contain is reduced, and the probability to find this sets becomes smaller: while the time required by I/O increases, the effectiveness of our computation strategy decreases.

Besides making parallel the computation of the distance matrix by using the blocks' strategy, we also used a multithreading strategy to compute each single block: in particular, we used a consumer-producer scheme [51], with C consumer threads and P producer threads. "Producer" threads scrolled through the input files, computing partial values for $\tilde{D}$ entries as they found sets of alignments with same search IDs; "consumer" threads stored these values in a binary tree, representing the matrix block, which is sparse. Navigating the binary tree is quite time consuming, leading to need much more consumers than producers.

Reasonable values of N, C and P strongly depends on the features of the supercomputer we use. To run this analysis, we used 10 nodes with 2 Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz (24 cores) and 768GB of RAM. We divided the input data in $N = 50$ groups, computing 1,275 blocks, each using $P = 3$ producer threads and $C = 20$ consumer threads. Each block computation occupied about 500 GB in RAM. The complete computation took about 30,000 CPU hours.

2.4.4 Metaclustering and merging

Once computed the distance matrix, the last steps of the protocol are not CPU intensive, but rather memory intensive, due to the amount of data we need to manage.

We compute metaclusters by Density Peak Clustering: the core of the method relies on calculating for each primary cluster i the local density, ρ_i and its minimum distance to points of higher density, δ_i (see Section 2.1). To compute these quantities we need to consider that the distance matrix to be used is stored in 1,275 files, of 2 GB each. First, we pre-allocate a vector for each quantity, ρ_i and δ_i. Both vectors are of size 2,353,198,020, such that the indexes correspond to the cluster ids, c_{id}, as defined in Section 2.4.2. Such vectors are sparsely populated, making the approach not efficient in terms of memory: on the other hand, this strategy allows for a fast access to the vectors' elements, and it improves significantly the code readability. We then loop through each file, saving the relevant information in the vectors while processed. Note that, since δ_i is a function of ρ_i, we need to calculate these quantities sequentially: we need then to loop twice through all distance matrix files. Once δ_i and ρ_i are defined we select density peaks using the maximum δ strategy, namely choosing as peaks all those primary clusters with $\delta = 1$ and $\rho = 1$ (see Section 2.1.1.2). We found 504,507 peaks, which have been then used to label the rest of the primary clusters based on the metacluster peak they have closer. If the minimum distance to a peak is bigger than $\mu_2 = 0.9$ we leave that primary cluster without classification. This step implies a third read of the distance matrix files.

Once defined metaclusters using DPC, we proceed to merge them (see Section 2.2.2.3). At this point we need to compute $D_{MC',MC''}$, namely the distance matrix between metaclusters, based on Equation 2.11. This computation does not involve a large number of operations: rather we have a memory constrain, since we need to handle a $504,507 \times 504,507$ matrix of "doubles" (8 bytes variables), corresponding to roughly 1TB of RAM (exploiting the fact that the matrix is symmetric). Luckily, with the resources we had we could load the whole matrix in RAM without need of further data partition. After the merging step we end up with $210,994$ metaclusters.

The output obtained so far is a list of primary clusters with their respective metacluster label, the last spanning from 0 to $210,993$.

Since the implementation is sequential we use only one HPC node with 2 Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz (36 cores) and 1536GB of RAM, also known as "fat nodes" because of their large RAM. The computation occupied approximately 1 TB of RAM and it took near 8 CPU hours to completion. The former is the biggest bottleneck of this step, the latter is negligible compared to the steps described in the previous sections.

2.4.5 Filtering

The result of the metaclustering procedure is a list of primary clusters, associated to a MC through a label. Tracing back each primary cluster to its search sequences, we end up with the final output of DPCfam: a list of protein regions $\mathcal{S}_i$ classified into $210,994$ metaclusters. The output is organized in a set of different files to simplify the navigation. As a last step we performed the filtering procedure, intended to remove outlier sequences, namely those that do not overlap with any other sequence in the same MC. To do this we first sort the output files with respect to the search ID. Then, we do a "stable sort" with respect to the MC label. In this way sequences of the same MC are stored contiguously, sorted with respect to the search ID. The sorting is done again using a C++ implementation of the radix sort algorithm. Thanks to this trick we just need to check, for each MC, if sequences with a given search ID overlap enough with any other with the same search ID.

If not, the sequence is removed. Since sequences are sorted with respect of the search IDs, it is easy to collect groups of sequences with the same search ID and do a fast comparison. We note that the remaining sequences in the MC having the same search ID should represent very similar regions of the same protein. This is a redundancy that has to be removed: of all ”good” sequences found in the MC with same search ID, we save the one that best overlaps with the average boundaries of such set. This procedure requires very few time, not affecting the total computing time of DPCfam.

Chapter 3

Results

We produced two implementations of the DPCfam protocol. The first one, also named DPCfam0, has been written and used for a proof-of-concept experiment, while the second one, tightly optimised and parallellized, has been used to run an all-to-all analysis of the UniRef50 database. In this chapter we will present the results obtained using the two implementations.

In Section 3.1 we describe the main measures we developed to investigate MCs' content and quality. To assess evolutionary consistency, we mostly refer to a ground truth (Pfam [2]) as a comparison. However, we know that such a ground truth should not be considered as an actual truth: indeed every classification we can compare with may contain errors and is prone to corrections. The measures we use here have been developed to take this into account: as we will see, it is often useful to integrate them with deeper analysis, using for example structural information.

In Section 3.2 we describe the results obtained by analysing small query datasets of protein sequences containing well-known Pfam families. Results obtained in this way have been deeply analysed, using structural-based information when available and other bioinformatics tools, such as HHpred [44] or DALI [52], to understand the nature of the clustered sequences and, when present, the reasons of inconsistencies between the classification resulting from our MCs and the Pfam classification. We also introduce some coverage measures, useful to understand how many known families have been identified by DPCfam, and how. Again, as a reference we use the Pfam database. These coverage measures are specific for small datasets as those presented in Section 3.2

In the Section 3.3 we present results obtained from the complete clustering of UniRef50. The number of MCs obtained here is orders of magnitude larger with respect to the previous analysis: it is beyond the scopes of this work to reach the same level of insight of metaclusters as we did before. Our results will finally be deposited in a database, which will be made accessible to other researchers for in-depth analysis. In order to analyze this large set of metaclusters we will introduce some broader coverage measures, to assess how many protein families DPCfam finds, and how. Again, we use Pfam as a reference.

3.1 Main measures to assess MCs' quality

To understand what kind of protein regions a MC includes and the quality of such grouping, we use two kinds of measures: absolute measures and ground truth related measures.

In the first case we gather general information about the clustered sequences: we focus on the sequences' length distribution and on the quantity of low complexity regions [53] contained in the sequences:

- A good MC should contain regions of about the same length: to asses this we compute the average length of each MC's sequences and its standard deviation. A MC with a large ratio between average length and its standard deviation makes us to flag it as possibly bad quality MC. In some cases, MCs may show a bimodal distribution: in this cases we expect such bimodality to be removed by profile-HMMs. When a profile-HMM is built, only the most conserved regions are considered for the model: this result in trimming the longer regions.

- Low Complexity regions (LC) are usually not frequent in families with a well-defined structural fold. This does not prevent sequences with many LCs to be part of a family, but still allow us to flag them as "questionable". For each sequence in a metacluster we search for LCs using the *segmasker* tool of NCBI BLAST+ [27]; we then compute the fraction of amino acids in LCs with respect to all amino acids collected in the MC. High values of this fraction makes us flag such an MC as a possibly bad quality MC.

As we will see, in the majority of the cases MC perform well with respect to these measures, showing coherence in their sequence's lengths and a very small quantity of LCs.

Ground truth related measures are more complicated. We will describe most of them in the following; while some other measures, specific to some particular case, will be introduced later.

3.1.1 Defining the Ground Truth of a protein region and of a MC

In order to compare MCs to any protein family database, we need to know the respective annotation of their protein regions. Such information is not always available, and in different situations we used different strategies to obtain it. Being Pfam the main ground truth we use in this thesis, in the following we will describe the procedure and the definitions to compare our classification to Pfam's.

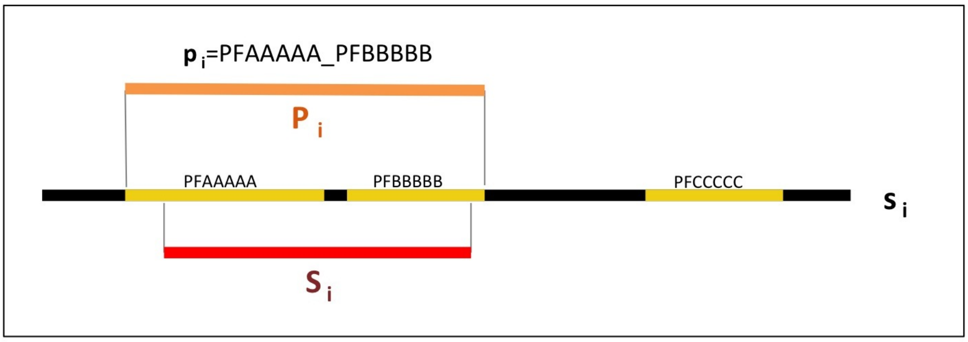


Figure 3.1: Schematic representation of Pfam ground truth architecture (GTA) assignment to a generic alignment $B_i(s_i, S_i)$. In this example, the full-length protein s_i has the following three-family architecture: PFAAAAA + PFBBBBB + PFCCCCC; the aligned region of the search sequence, $\mathcal{S}_i$, instead covers (partially) only PFAAAAA and PFBBBBB; thus, the Pfam ground truth of B_i is p_i=PFAAAAA_PFBBBBB (note that a 1-residue overlap of $\mathcal{S}_i$ with a family is enough for the latter to be included into the GTA); in orange we show $\mathcal{P}_i$, that is, the full region covered by the GTA families on the sequence s_i.

Let us assume that we know the Pfam classification for every protein collected by our MCs: in particular, we know the Pfam annotation of the respective protein sequences. We define the Pfam Ground Truth Architecture (GTA) p_i of a region $\mathcal{S}_i$ as the ordered set of Pfam families that overlap with $\mathcal{S}_i$, if any. The order of the families reflects their relative position along $\mathcal{S}_i$. For example, suppose that we want to determine the GTA of the region of protein s_i =Q5BH58 spanning positions 132 to 567. Pfam annotation for Q5BH58 is as follows: PF02190 (aa 10-258), PF00004 (aa 482-625), PF05362 (aa 706-915). In this case, the GTA of $\mathcal{S}_i$ is represented by p_i =PF02190_PF00004, comprising all Pfam families having at least a one-residue overlap with this region. We can alternatively define the GTA in terms of Pfam clans to which each Pfam family is associated (in this case $p_i(clan)$ =CL0178_CL0023); again, the GTA is an ordered string of (clan) ids. If a family is not associated to a clan in Pfam, we use the family id in the clan GTA. The boundaries of the subregion of $\mathcal{S}_i$ covered by the GTA p_i will be indicated as $\mathcal{P}_i$. If p_i includes two or more families, $\mathcal{P}_i$ will span also non-annotated residues between those families, if present (see Figure 3.1). In the example above the $\mathcal{P}_i$ of $\mathcal{S}_i$ is the interval between residue 132 and 567.

Next, we define the Pfam Dominant ground truth Architecture (DA) of a metacluster as the most abundant GTA among all of its member regions. The DA can be defined both at a family or at a clan level.

3.1.2 Comparing MCs with their ground truth: evolutionary consistency

When using Pfam annotations to analyze the evolutionary consistency of our MC classification, we take into account the following: i) evolutionary distances between families within a Pfam clan can differ greatly; in particular, some families may be very closely related to each other. For this reason, it is often more informative to look at consistency of annotation in MCs at the *clan* level (see Section 1.5.2); ii) along with many full-length sequences, UniRef50 also contains sequence fragments. This may be relevant when comparing MC member annotations, especially for those MCs associated to multi-family DAs. iii) Pfam classification of families and clans can be incomplete; as a consequence, regions in UniRef50 that are not currently annotated in Pfam may still belong to known Pfam families and clans.

Given a MC, we first determine its DA both at the family and at the clan level, and we indicate, respectively, %DA and %DAC (C=clan) their relative percentages among MC members. Hereafter, we call "DA members" those for which, at the clan level, have the DA as Ground Truth Architecture. Next, we consider MC members that match the DA (again, at the clan level) only partially. While this makes sense in light of observations (ii) and (iii) above, it also allows for some variability in length among MC members. We compute the percentage of MC members with a GTA that lacks one or more of the DA clans but, at the same time, doesn't feature any extra clan(s). We sum this percentage to %DAC and report it as %DACF (F=fewer); we still ask that the remaining clans are in the same order as in the DA. Note that MC members lacking any Pfam annotation are counted in %DACF. This is consistent with the idea that having no Pfam annotation does not imply that a region is not part of an existing Pfam clan (observation (iii) above). Finally, we compute the percentage of MC members with a GTA that features one or more Pfam clans not

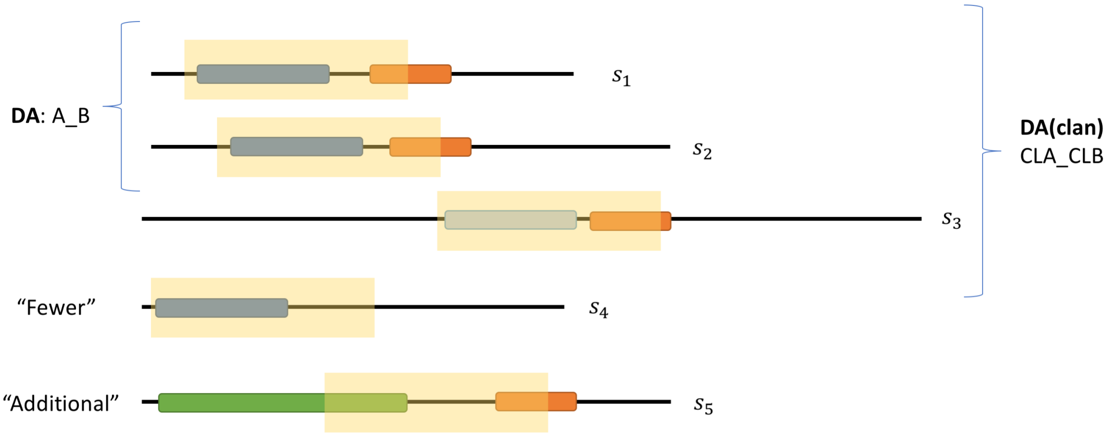


Figure 3.2: Graphical explanation of %DA, %DAC, %DACF and %DACFA on a toy MC containing 5 sequences. Yellow boxes represent the clustered region on proteins, the latter being s_1, s_2, s_3, s_4 and s_5. Coloured blocks represent Pfam families, where the same color indicate the same family, while lighter/darker shades of the same color indicate families in the same clan.

found in the DA but, at the same time, contains at least one of the original DA clans. We sum this to %DACF and call it %DACFA (A=additional). In Figure 3.2 we give a graphical representation of this procedure on a toy MC containing 5 regions. In this MC, the DA at the family level would be A_B (A: blue block, B: orange block), with %DAF=40. At the clan level, also region on protein s_3 would be included, being the light blue block of the same clan of A: the DA(clan) would be CLA_CLB and then %DAC=60. Region on protein s_4 lacks of the orange block, and therefore would be counted in the %DACF computation, so that %DACF=80. Finally, the last region includes an extra family (green) in place of A (blue), and still captures a portion of B (orange) family. Such a sequence will be counted in the %DACFA computation, so that %DACFA=100. Note that if s_5 didn't contain the orange block, this region could not be considered as "additional" and the final value of %DACFA would have been 80.

%DA or %DAC close to 100% indicate that, according to Pfam, most members of the metacluster share a homologous core region that covers all families or clans in the DA. As we will see, the analysis of differences between these percentage scores greatly facilitates the identification of MCs that may not be evolutionarily sound, as well as those MCs that may help improving the Pfam classification by expanding family and clan membership, by uncovering novel domains or by pointing to potential inconsistencies in the existing annotation. Schematically:

- Differences between %DA and %DAC tell us to which extent member sequences are spread out across multiple families pertaining to the clan(s) represented in the DA.

- Large increases from %DAC and %DACF can point to MCs with the potential to increase coverage of existing Pfam families or clans.

- Large increases in %DACF and %DACFA can be a sign of an incomplete Pfam annotation for members of the families in the DA.

- Low %DAC and high %DACF indicate MCs constituted of member sequences that are devoid of any Pfam annotation: such MCs are good candidates for protein family discovery.

Results reported in Section 3.2 will give more insights to justify and explain these affirmations.

3.1.3 Comparing MCs with their ground truth: boundaries

Comparison between the DPCfam and Pfam classifications cannot be reduced to the percentage of families and clans among MC members. The degree of agreement between the boundaries of a MC's region $\mathcal{S}_i$ and the boundaries of the respective Pfam annotation $\mathcal{P}_i$ is also important.

We here introduce two measures, namely F_{red} and F_{ext}, which allow quantifying boundaries agreement.

$$F_{red,i} = \frac{|\mathcal{P}_i| - |\mathcal{S}_i \cap \mathcal{P}_i|}{|\mathcal{P}_i|} = \frac{|\mathcal{P}_i| - |\mathcal{I}_i|}{|\mathcal{P}_i|} \tag{3.1}$$

$$F_{ext,i} = \frac{|\mathcal{S}_i| - |\mathcal{S}_i \cap \mathcal{P}_i|}{|\mathcal{S}_i|} = \frac{|\mathcal{S}_i| - |\mathcal{I}_i|}{|\mathcal{S}_i|} \tag{3.2}$$

$F_{red,i}$ represents the fraction of the Pfam region $\mathcal{P}_i$ that is not covered by the clustered region $\mathcal{S}_i$; vice versa, $F_{ext,i}$ is the fraction of the region $\mathcal{S}_i$ that is not covered by the Pfam region $\mathcal{P}_i$ (see Figure 3.3 A).

We compute these quantities using the MC's Dominant Architecture (DA) rather than the regions' Ground Truth Architecture (GTA), more specifically using the DA at a clan level. We restrict the computation only to those MC's regions that are actually DA members: therefore, to characterize boundaries of an entire MC with respect to its DA we compute the averages of $F_{red,i}$ and $F_{ext,i}$ on the subset of the DA members. We denote these averages as F^{MC}_{red} and F^{MC}_{ext}.

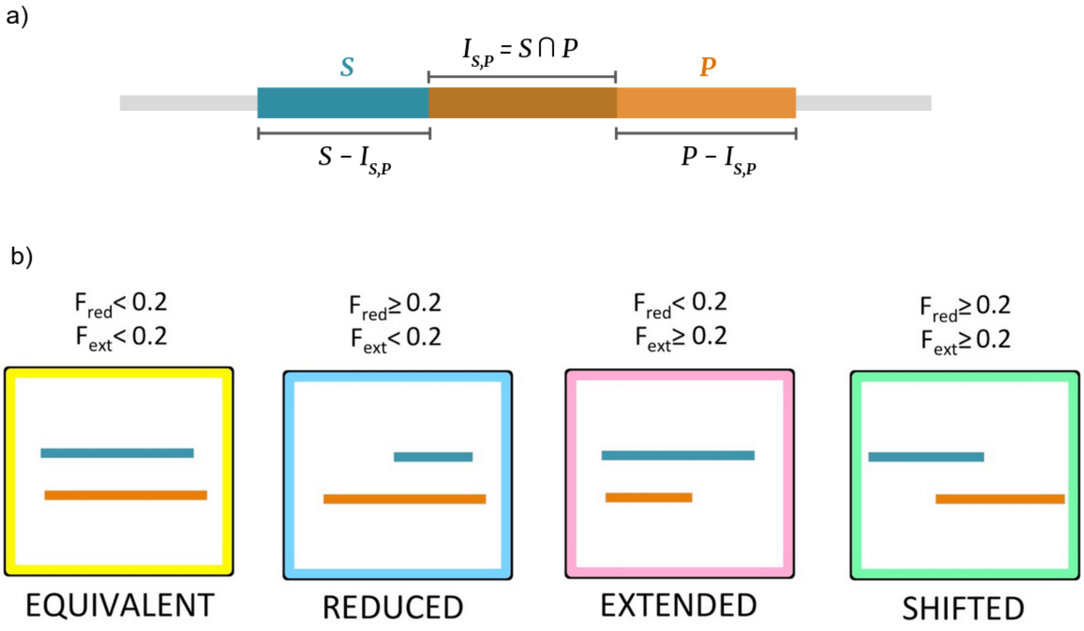


Figure 3.3: A) Graphical representation of the elements used to compute F_{red} and F_{ext} of a protein region $\mathcal{S}$ (blue) with respect to its ground-truth associated region $\mathcal{P}$ (orange). B) Pictorial representation of the four categories of metaclusters we defined based on F^{MC}_{ext} and F^{MC}_{red} values.

To provide some qualitative insight, we classify MCs into the following four categories according to the agreement of their DA members with the DA's boundaries, developing a color-code that will be used in the following tables and graphs (see Figure 3.3 B):

- *equivalent* : both F^{MC}_{ext} and $F^{MC}_{red} < 0.2$, yellow;
- *reduced* : $F^{MC}_{ext} < 0.2$ and $F^{MC}_{red} \geq 0.2$, blue;
- *extended* : $F^{MC}_{ext} \geq 0.2$ and $F^{MC}_{red} < 0.2$, pink;
- *shifted* : both F^{MC}_{ext} and F^{MC}_{red} are ≥ 0.2, green.

3.2 Proof-of-concept on PUA and P53 clans

As a proof-of-concept for DPCfam, we analysed two small query sets derived from two Pfam clans, PUA [11] and P53-like [12]. Such datasets contain a limited number of proteins (∼2,000 to ∼4,000), representative of proteins of the respective clan. The procedure to obtain one of these datasets is the following: given a Pfam clan, for example PUA, we generate a dataset constituted of all UniRef50 full-length sequences that carry a PUA clan member annotation. This annotation is obtained by matching UniProtKB sequences' identifiers with those of sequences in Pfam-A.full v. 31, namely Pfam anotation of Reference Proteomes. The dataset obtained is then named PUA_UR50, containing 4,083 protein sequences. With the same procedure we obtain P53_UR50, containing 2,022 protein sequences.

3.2.1 Defining the Pfam Ground truth

In Section 3.1.1 we explained how we assign a GTA to a protein region contained in a MC. Such procedure requires to know the Pfam annotation of the respective full-length protein; however, obtaining this is not straightforward. As previously mentioned, our protein database of reference is UniRef50 (v. 2017_07). Not all sequences in UniRef50 are annotated in Pfam, thus we are not able to use the Pfam database family assignments directly to inspect the MCs' content. Pfam offers annotation of the UniprotKB database, which still does not cover entirely UniRef50. Therefore, in this analysis we decided perform a *de novo* annotation of the MCs' proteins, following the Pfam method. We first collect all the full-length proteins whose regions appear in our MCs. Then we ran each protein sequence against the set of all Pfam_A.hmm models (v. 31) using the hmmscan program from the HMMER 3.1b2 suite [54], and we assign to each protein sequence a Pfam family architecture according to the models' manually-curated gathering thresholds. In the case of multiple significant matches overlapping along the same protein sequence, we keep only the Pfam annotation corresponding to the lowest E-value. Overlaps are calculated using start and end alignment positions. Note that this protocol does not allow domain nesting.

3.2.2 Clustering of proteins from the PUA clan

Starting from the PUA_UR50 query dataset, our clustering method produces 71 MCs in total (Figure 3.4) for the MC size distribution). We find 19 MCs mapping to PUA families (Table 3.1) and 52 mapping to PUA associated families (Table 3.2). As previously mentioned, MCs can represent single or multi-family architectures and their DAs may or may not contain PUA clan families. Also, different MCs can map to the same Pfam family or architecture.

MC	DA	% DAF	%DAC	%DACF	%DACFA	F_{ext}^{MC}	F_{red}^{MC}	Extra Clans
1	PF02190	81.3	81.3	83.4	98.9	0.03	0.03	1
2	PF02190_PF00004_PF05362	62.2	62.2	95.2	100.0	0.00	0.14	
3	PF04146	98.5	98.5	99.0	99.9	0.07	0.03	
4	PF04266	69.1	69.1	99.6	99.7	0.16	0.01	
5	PF04266	98.4	98.4	100.0	100.0	0.05	0.02	
6	PF17126_PF13636	47.1	47.1	100.0	100.0	0.00	0.16	
7	PF17125_PF01189_PF17126_PF13636	42.1	42.1	99.5	100.0	0.00	0.01	
8	PF01878	97.7	97.7	98.0	99.5	0.02	0.03	
9	PF14306_PF01747	85.1	85.1	96.5	99.6	0.00	0.08	
10	PF04266	62.9	62.9	99.2	99.2	0.02	0.03	
11	PF06221_PF04266	71.7	71.7	98.3	100.0	0.13	0.00	
12	PF03657	70.6	71.6	100.0	100.0	0.03	0.06	
13	PF04266	86.0	86.0	100.0	100.0	0.05	0.61	
14	PF04146	99.7	99.7	100.0	100.0	0.01	0.66	
15	PF02190_PF03226	26.5	26.5	99.7	99.7	0.00	0.52	
16	PF14306	98.1	98.1	100.0	100.0	0.01	0.36	
17	PF02190	99.4	99.4	100.0	100.0	0.08	0.51	
18	PF01472	38.4	38.4	75.0	98.5	0.47	0.01	2
19	PF02190	94.4	94.4	99.4	100.0	0.58	0.78	

Table 3.1: DA annotation of PUA_UR50 MCs containing PUA families. For each MC, we report: the family-level Pfam Dominant ground truth Architecture (DA); the percentage of members featuring a DA annotation either at the family (%DAF) or at the clan (%DAC) level; %DAC plus the percentage of members lacking one or more of the DA clans but having no additional clan's annotation (%DACF); %DACF plus the percentage of members having clans outside of the DA but at least one DA clan (%DACFA); for DA members, the average extent of the overlap with the DA, F_{ext}^{MC}, F_{red}^{MC}; the number of extra clans that feature in %DACFA (only those present in at least 5% of clan members). MCs are colored according to the overlap between DA members and DA annotation: equivalent (yellow), reduced (blue), extended (pink) and shifted (green) (see Sections 3.1.2 and 3.1.3 for definitions).

MC	DA	% DAF	%DAC	%DACF	%DACFA	F_{ext}^{MC}	F_{red}^{MC}	**Extra Clans**
A1	PF00069	82.8	98.8	98.8	99.9	0.02	0.15	
A2	PF13561	21.2	31.8	75.7	100.0	0.04	0.12	
A3	PF13923	23.6	97.1	99.6	99.6	0.15	0.02	
A4	PF00004	77.3	96.8	97.4	100.0	0.15	0.05	
A5	PF01189	75.3	75.6	75.8	100.0	0.13	0.10	
A6	PF00270	70.9	73.9	99.4	99.9	0.17	0.01	
A7	PF13181	6.6	33.7	87.4	99.8	0.15	0.10	
A8	PF00271_PF04408	56.6	56.6	62.4	100.0	0.09	0.09	1
A9	PF00083	70.4	94.6	94.8	100.0	0.07	0.10	
A10	PF08282	99.1	99.1	99.1	100.0	0.02	0.02	
A11	PF05362	83.5	97.0	98.6	99.7	0.06	0.10	
A12	PF13302	99.7	99.7	99.7	100.0	0.09	0.00	
A13	PF01509_PF16198	53.0	53.6	80.0	99.8	0.13	0.02	1
A14	PF00696	51.9	51.9	52.0	99.8	0.09	0.01	1
A15	PF01583	98.1	99.1	99.1	99.9	0.08	0.01	
A16	PF01507	98.3	98.3	99.2	100.0	0.08	0.04	
A17	PF04408_PF07717	84.4	84.4	99.7	100.0	0.16	0.10	
A18	PF13187	23.7	71.8	73.2	89.6	0.15	0.09	
A19	PF04752	99.1	99.1	99.1	100.0	0.03	0.03	
A20	PF00719	99.0	99.0	99.2	100.0	0.13	0.03	
A21	PF10343	99.0	99.0	99.0	100.0	0.13	0.07	
A22	PF00383	93.7	94.7	99.5	99.5	0.20	0.13	
A23	PF06221	99.4	99.4	100.0	100.0	0.03	0.16	
A24	PF13744	58.6	98.1	99.4	99.4	0.08	0.16	
A25	PF00076	62.3	62.3	99.3	99.3	0.16	0.00	
A26	PF02470_PF02470	68.9	68.9	99.3	100.0	0.07	0.01	
A27	PF00067	97.1	97.1	97.5	100.0	0.01	0.37	
A28	PF01189	66.7	66.9	66.9	100.0	0.19	0.62	
A29	PF05958	45.2	98.9	99.0	100.0	0.01	0.55	
A30	PF01509_PF16198	74.7	74.7	100.0	100.0	0.00	0.75	
A31	PF07728	13.1	17.0	100.0	100.0	0.18	0.21	
A32	PF00083	79.3	100.0	100.0	100.0	0.01	0.37	
A33	PF13733_PF02709	97.6	97.6	99.1	100.0	0.08	0.20	
A34	PF10539	100.0	100.0	100.0	100.0	0.01	0.21	
A35	PF07728	8.1	11.2	100.0	100.0	0.17	0.61	
A36	PF00011	100.0	100.0	100.0	100.0	0.00	0.40	
A37	PF08423	91.4	91.4	91.4	100.0	0.07	0.24	1
A38	PF07728	98.0	98.3	99.9	99.9	0.42	0.10	
A39	PF00642	27.6	31.3	77.9	99.7	0.69	0.14	
A40	PF10672	93.1	97.3	97.7	99.8	0.37	0.00	
A41	PF01702	98.8	98.8	99.5	100.0	0.37	0.08	
A42	PF13516_PF13516	20.1	20.6	60.4	97.2	0.55	0.00	
A43	PF06221	93.3	93.3	99.6	100.0	0.70	0.01	
A44	PF00642	92.3	92.3	92.8	100.0	0.27	0.00	
A45	PF00011	88.2	88.2	90.3	100.0	0.53	0.06	
A46	PF01253	84.3	84.3	87.6	100.0	0.75	0.02	1
A47	PF02195	99.1	99.1	100.0	100.0	0.58	0.03	
A48	PF00004	6.2	7.2	99.0	99.0	0.89	0.91	
A49	PF01509_PF16198	50.1	50.1	58.2	99.5	0.36	0.65	1
A50	PF01583	99.6	99.6	99.6	100.0	0.21	0.57	
A51	PF07728	5.1	5.1	100.0	100.0	0.91	0.90	
A52	PF00226	94.4	94.4	97.6	97.6	0.62	0.62	

Table 3.2: DA annotation of PUA_UR50 MCs containing PUA associated families. MCs are labeled with "A" prefix (as "associated"). (See Table 3.1).

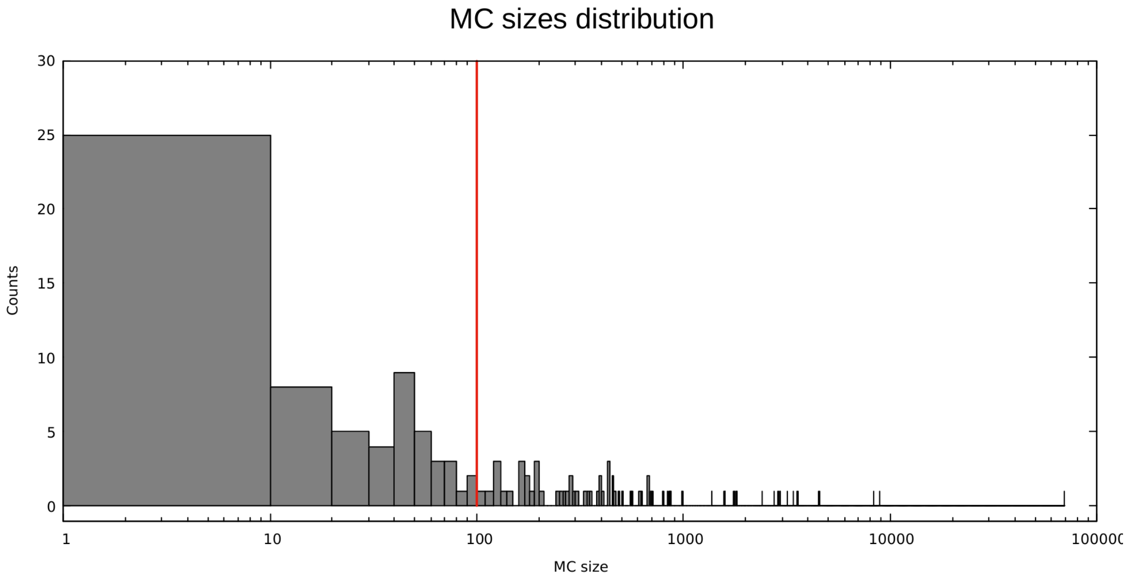


Figure 3.4: Size distribution of PUA_UR50 metaclusters after redundancy reduction at 95% sequence identity. Here, we include also MCs with less than 100 elements.

3.2.2.1 Evolutionary consistency of MCs

The first question we address is whether DPCfam-generated MCs are evolutionarily consistent. In other words, we ask if MCs are formed of member sequences that share a core homologous region and could thus potentially be used as seeds for building protein families. In Tables 3.1 and 3.2 we report the percentage of member regions with a GTA that matches exactly (%DAF - family level and %DAC – clan level) or partially (%DACF and %DACFA) the DA of the cluster. %DAF or %DAC close to 100% indicate that, according to Pfam, most member sequences share a homologous core region that covers all families or clans in the DA. For example, 99.7% of 1,795 MC-A12_PUA member regions are annotated in Pfam as Acetyltransf_3 (PF13302). Overall, 43.7% of MCs have %DAC>95%. Differences between %DAF and %DAC can tell us to which extent member sequences are spread out across multiple families pertaining to the clan(s) represented in the DA. The number of Pfam families and clans and their relative weight within an MC can be better appreciated from the graphical representation in Figure 3.5 (for MCs with >500 members). For instance, MC-A3_PUA maps to several different families within the RING (CL0229) clan. This is not surprising given that the Pfam evolutionary profiles of zinc finger families within the RING clan tend to overlap (see e.g. the E-values of the families' profile-profile alignments in the clan's “Relationships” tab on the Pfam webserver).

When we add to %DAC all those members with a GTA matching only partially

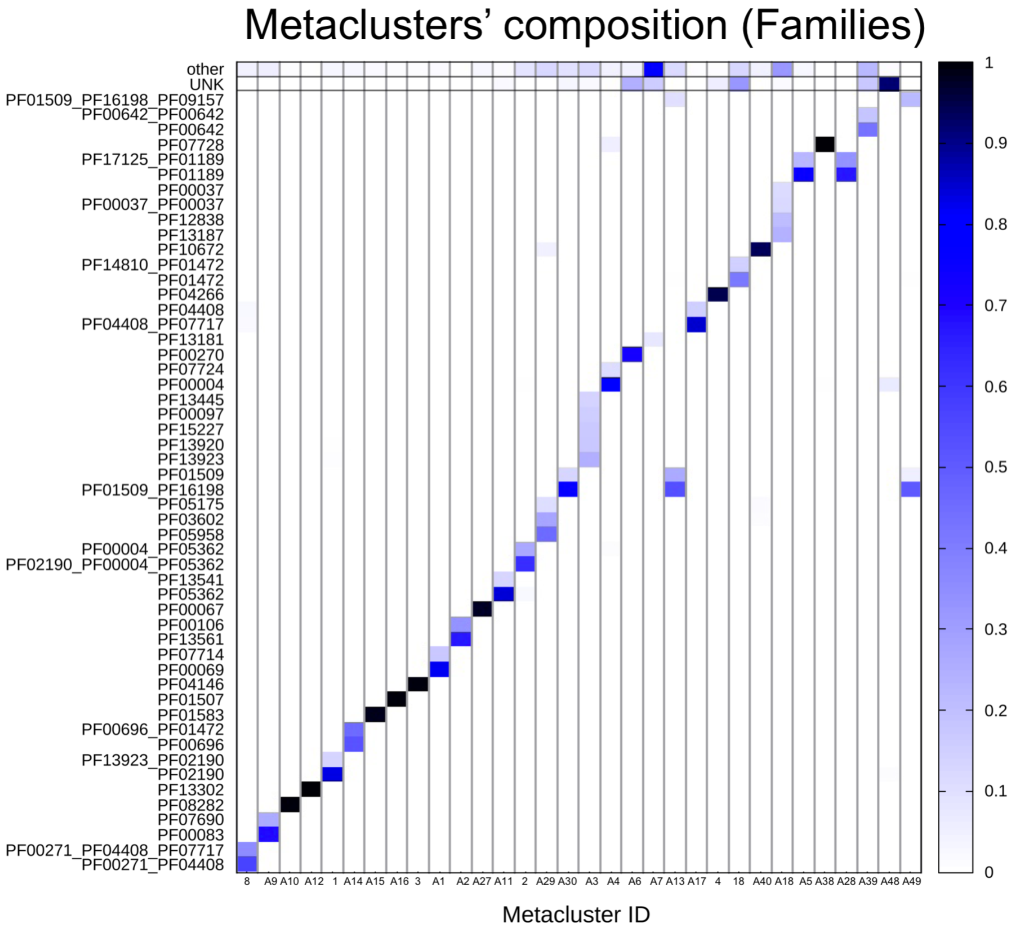


Figure 3.5: PUA_UR50 MCs vs Pfam annotation. On the x-axis, we list the 32 MCs (both PUA and non-PUA) with more than 500 member regiones; on the y-axis we list the Pfam GTAs (family level) represented in each MC. We report only GTAs mapping to at least 10% of MC members and we aggregate all the remaining ones under the label ”other”; finally, we label “UNK” MC members with no Pfam annotation. The heatmap is colored according to GTA fraction.

the DA of the MC (%DACF and, finally, %DACFA) we achieve close to full coverage in most MCs. Indeed, only one MC (MC-A18_PUA) has %DACFA<90 (Figure 3.6 and, again, Table 3.2).

Large percentage increases in the %DACF and %DACFA columns can point to MCs with the potential to increase coverage of existing Pfam families or clans. For example, metaclusters MC-A6_PUA and MC-4_PUA feature rather large increases in %DACF (25.7% and 30.5%, respectively). Given that the DA of these MCs is single-domain, such increases correspond to the percentage of member regions lacking any annotation in Pfam. MC-A6_PUA DA is composed of the helicase family DEAD (PF00270). Unannotated MC-A6_PUA member regions are almost always

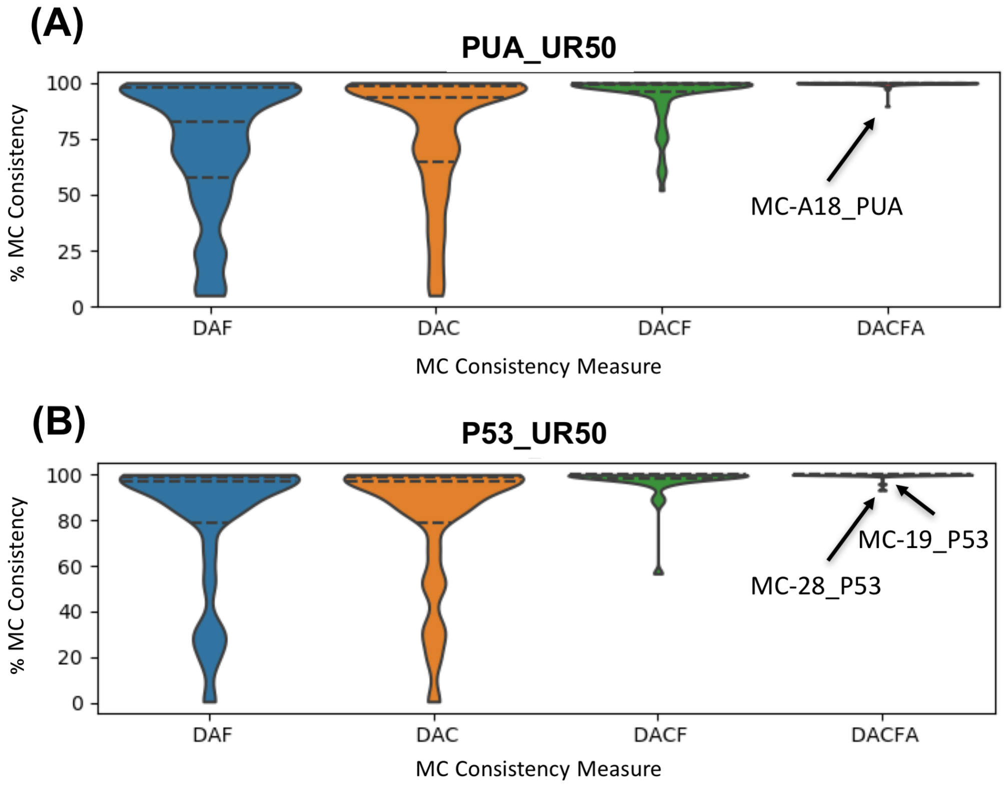


Figure 3.6: Violin plots of the distribution of %DAF, %DAC, %DACF and %DACFA. See Section 3.1.2 for definitions. (A) MCs generated from the PUA_UR50 dataset (B) MCs generated from the P53_UR50 dataset. Violin plots are such that their width is proportional to the fraction of MCs with a given value of the respective consistency measure. We label outlier MCs for %DACFA.

found at the N-terminus of proteins with one or more families in the Helicase_C + HA2 + OB_NTP_bind architecture. Since this is a common Pfam architecture for the DEAD domain, unannotated regions in MC-A6_PUA are likely to represent yet unrecognized members of the DEAD family. The DA of MC-4_PUA, instead, corresponds to the ASCH domain (PUA clan), with about 69% of member regions carrying this Pfam annotation. While the vast majority of remaining regions are not annotated in Pfam, in InterPro many carry an ASCH/PUA-related annotation. A closer examination reveals that MC-4_PUA is constituted of regions that are part of the "ASC-1 proper family", as defined in the work by Iyer et al. [55], in which ASCH domains were defined for the first time. The "ASC-1 proper family" was there characterized as having a long insertion between the 3rd and 4th strand of the ASCH fold. Now that structures are available for this particular ASCH subfamily, we can additionally recognize that the domain as originally defined was cut slightly short at the C-terminus, excluding a final, extra strand and short alpha-helix (see Figure 3.7). The presence of PDB structures for the C-terminal ASC-1 domain of human activating signal cointegrator 1 protein (e.g. 2E5O, 5Y7D), allowed us to build an alignment covering the whole structural domain that the family represents. Using this alignment to build a profile-HMM and running it against the Reference Proteomes database appears to capture a good number of yet unannotated regions. A large increase in %DACFA can similarly be a sign of an incomplete Pfam annotation for members of the families in the DA. One example is likely to be MC-A8_PUA, in which several member regions are likely to lack annotation for the C-terminal domain OB_NTP_bind - PF07717.

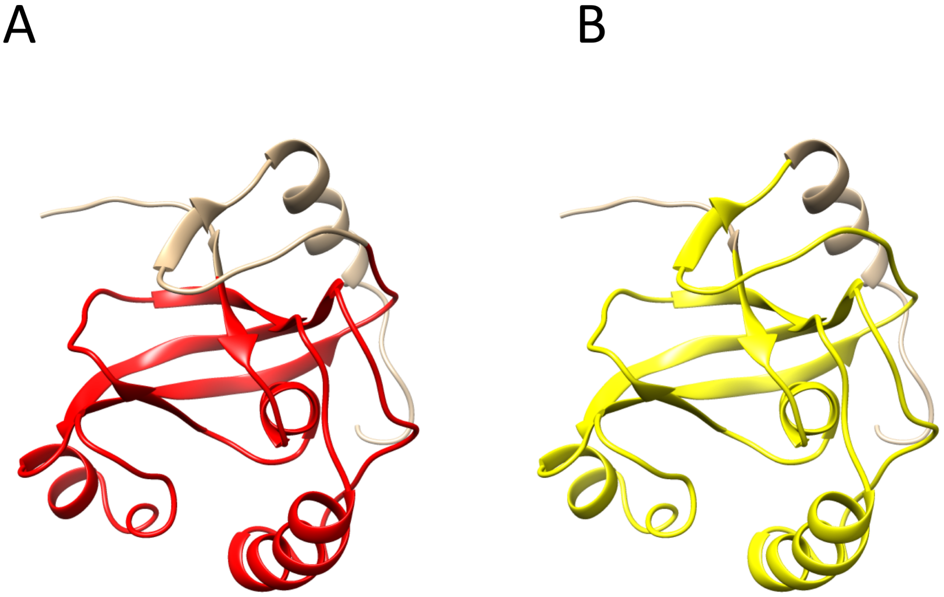


Figure 3.7: Structure of PDB protein chain 2e5o_A (NMR model #0.1). (A) We highlight in red the Pfam ASCH annotation, roughly corresponding to the boundaries of the "ASC-1 proper family" as found in Iyer *et al.*. It can be seen that the final strand-helix motif is missing from the annotation. (B) We highlight in yellow the region captured by MC-4_PUA profile-HMM, which captures the whole ASCH region, plus the extra strand. Annotation according to Pfam v32.0.

In other instances, percentage increases in the %DACF and %DACFA columns are not due to incomplete Pfam annotation but rather to the presence of subgroups of MC members featuring radically different lengths. Two such examples are MC-A14_PUA and MC-2_PUA (Figures 3.8 A and B). In these cases, differences in annotation between members could be easily resolved, for example, by trimming the respective MSA alignments to the shortest lengths. Further, there are cases in which the DA does not provide an accurate description of the annotation of the MC. This happens when a family has only a marginal overlap with a number of member regions and absolutely no overlap with others. In this case, we can have large increases in %DACF or %DACFA that are artifacts of the way we annotate the GTA of MC members.

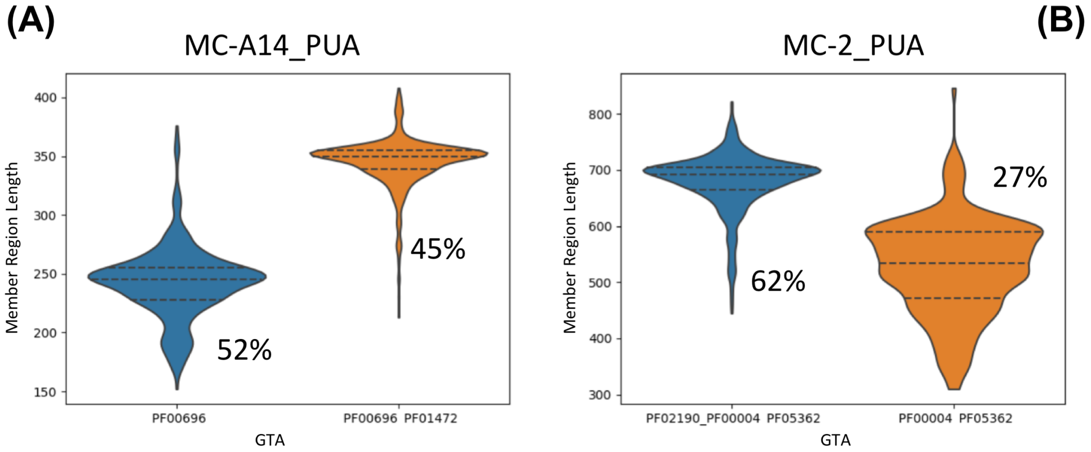


Figure 3.8: Distribution of member regions' length for MC-A14_PUA (A) and MC-2_PUA (B). For each plot, we show the distribution of lengths of DA regions (i.e. matching the DA exactly) (blue) and, additionally, of those matching the second most abundant GTA in the MC (orange); we report the percentage of members with a given architecture next to each violin plot (note that percentages need not sum to 100%).

One example is MC-A28_PUA where about 33% of member regions overlap with a small portion of family PF17125, which is located at the N-terminus and is not part of the DA of the metacluster. We note that in principle it would be possible for members counted as part of %DACFA to map to completely different, non-overlapping sections of the DA. These would be regions that are not homologous to each other. During our analysis of the PUA clan and associated families, however, we did not come across any such example, suggesting that these are unlikely to be common occurrences.

A quick look at Table 3.1, reveals a couple of outstanding cases among all MCs produced. First MC-A18_PUA, which is by far the metacluster with the lowest %DACFA (89.6%). This indicates that >10% of member regions carry Pfam annotation that appears to be incompatible with the DA of the MC; in other words, these regions would appear to be evolutionary unrelated to the others. The metacluster's

DA is constituted of Pfam family Fer4 9 (PF13187), which itself is part of the 4Fe-4S (CL0344) clan. Most families in this clan represent iron-sulfur cluster binding motifs (Fe-S BMs) characterized by a CxxCxxCxxxC signature and they often feature two consecutive copies of such motif, PF13187 being one of them. A more than 45% increase in member coverage from %DAF to %DAC for MC-A18_PUA indicates that Pfam annotation of MC members covers other families of the 4Fe-4S clan. Examples are members annotated as part of the double-motif families Fer4_7 (PF12838) and Fer4_10 (PF13237) as well as those annotated with two copies of the single motif family Fer4 (PF00037). All of the above are families with close evolutionary relationships within the clan. There is, however, a fraction of members that are annotated as belonging to families such as Radical_SAM (PF04055) and DUF362 (PF04015) that are found in clans other than CL0244. What is happening in these cases, however, is that the MC members span Fe-S BM regions that are nested within these longer domains. As mentioned above, our in-house Pfam annotation protocol does not take into take into consideration nested domains. If family A spans region a to b of a protein and family B the region a' to b' of the same protein with a'>a and b'<b, region a'-b' is assigned to one family only, the one with the lowest E-value, which will generally belong to the longest family. This is what happens for some of the MC-A18_PUA members, whereby regions that in Pfam are annotated as Fe-S BMs are instead annotated by our protocol as belonging to the family the BMs are nested within; we show one example of this in Figure 3.9. In conclusion, we can say that the vast majority of MC-A18_PUA member sequences consistently represent regions spanning Fe-S BMs.

A second outstanding case is represented by MC-18_PUA. This metacluster has a significant number of member regions that feature extra clans not part of the DA represented by the PUA (PF01472) family (>20% increase in %DACFA). While other clans have even larger %DACFA increases, MC-18_PUA is unique in that it is the only one featuring two extra non-DA clans (last column in Table 3.1).This would not constitute a problem if the clans were added sequentially along the sequence but could be problematic if the two clans were found in a similar position upstream or downstream of the DA in different MC members. For this reason, MC-18_PUA needs to be analysed in detail. We start by observing that $F_{ext}^{MC} = 0.47$, indicating that even DA members typically extend well beyond their PUA domain although in a (in this case N-terminal) region not annotated by Pfam. A number of other MC members, however, feature additional Pfam annotation at the N-terminus of the PUA domain: 13.6% feature a TGT_C2 (PF14810) domain, 8.4% a DUF1947 (PF09183)

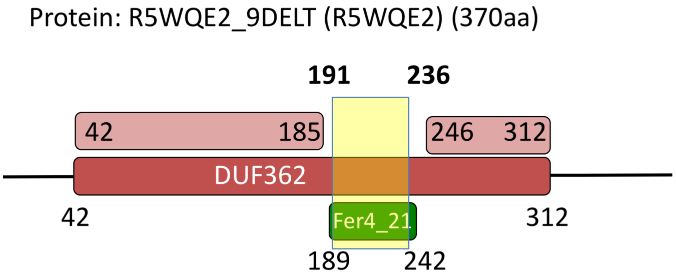


Figure 3.9: Example (protein R5WQE2) of nesting of an MC-A18_PUA region into a family of the “DUF362-like superfamily” - CL0471 clan. Solid-colored rectangles mark Pfam family annotations. The light red rectangle shows the region of R5WQE2 that aligns to the DUF362 profile-HMM. Note that in this specific case, even in the Pfam annotation nesting of Fer4_21 into DUF362 is not accounted for. The yellow box marks a hit obtained using the profile-HMM derived from MC-A18_PUA.

domain and, finally, 1% a TruB_C_2 (PF16198) domain. When present, these families are well covered by the MC-18_PUA member sequences: 97% of TGT_C2 amino acids are covered, 67% of DUF1947 and 61% of TruB_C_2, respectively. Worryingly, these three families are not found in the same Pfam clan, that is, they are not recognized as homologous by the Pfam classification: DUF1947 is part of the pre-PUA (CL0668) clan that, as the name indicates, is constituted of regions that are found N-terminal to PUA domains, TruB_C_2 is part of the PseudoU_synth (CL0649) clan and, finally, TGT_C2 is not part of any Pfam clan. We notice, however, that TGT_C2 regions are almost always found N-terminal to PUA domains; more importantly, alignment between representative structures of TGT_C2 and DUF1947 reveals striking similarities (see Figures 3.10 and 3.11) thus suggesting a common evolutionary origin for the two families. TGT_C2 would then represent a novel pre-PUA domain to be added to the Pfam clan of the same name. Interestingly, even a very sensitive profile-profile alignment method such as HHpred [44] appears not to be able to find a relationship between TGT_C2 and pre-PUA. In particular, when we ran HHpred using the Pfam seed multiple sequence of family TGT_C2 against Pfam v33.1 we found no significant match to any of the pre-PUA clan families. The other extra family found in about 1% of MC-18_PUA member regions, TruB_C_2, is instead structurally (thus evolutionarily) unrelated to both DUF1947 and TGT_C2. Indeed, most MC-18_PUA alignments that feature TruB_C_2 have E-values of borderline significance (>0.01) further supporting the notion that these are likely to represent noise.

In summary, analysis performed using Pfam annotation suggests that the vast majority of MCs are evolutionarily sound with member sequences that share between them a core homologous region. This core region may correspond to the DA of the metacluster or be longer/shorter as we will discuss more in detail in the following.

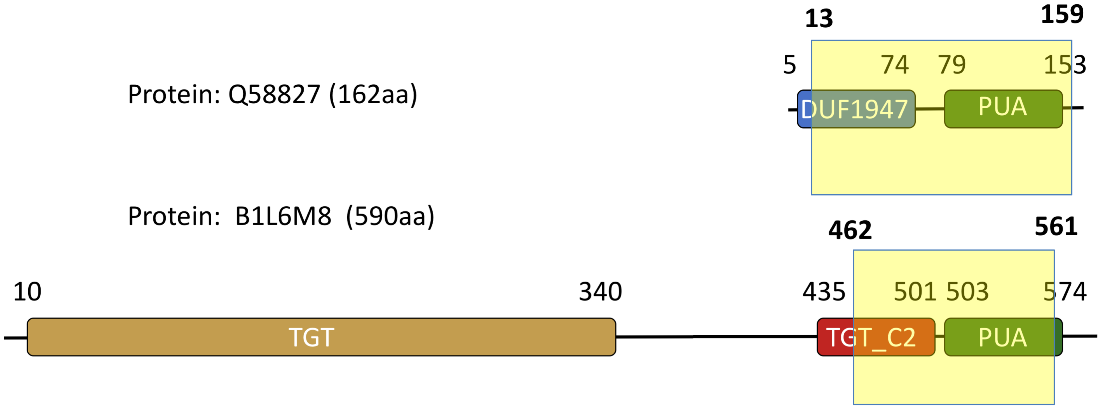


Figure 3.10: Annotation for protein Q68827 and B1L6M8 (A) Top: Pfam annotation for Q68827; the yellow box indicates a hit obtained using profile-HMMs derived from the metacluster MC-18_PUA. Bottom: Pfam annotation of protein B1L6M8; the yellow box indicates a hit obtained using profile-HMMs derived from the metacluster MC-18_PUA.

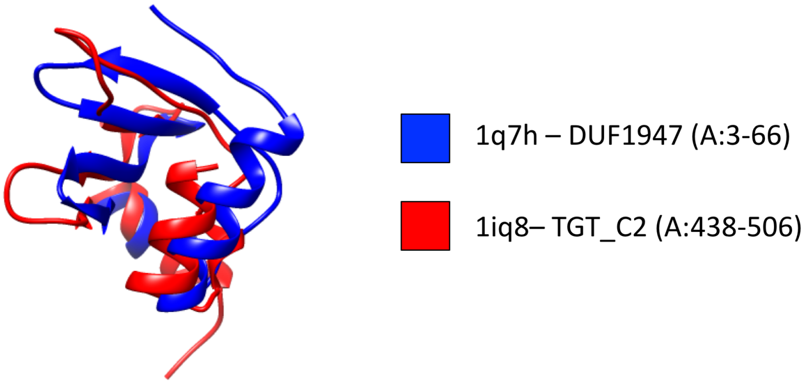


Figure 3.11: Structural alignment of DUF1947 domain is pdb structure 1q7h (A:3-66) (protein Q9HIB8) with TGT_C2 domain in pdb 1iq8 (A:438-506) (protein O58843). Aligned with Dali (Holm, 2019) ; Z 4.5 , rmsd 3.0, nres 60 and %ID 12 .

MC (PUA)	Size	Average Length	SDL	LC fraction	MC (PUA)	Size	Average Length	SDL	LC fraction
1	1575	207.4	42.5	0.04	*11*	120	396.3	*61.8	0.04
2	862	623.1	*102.9	0.04	*12*	109	154.2	26.7	0.02
3	791	152.7	30.5	0.02	*13*	430	40.6	3.3	0.00
4	682	125.1	16.9	0.01	*14*	399	54.2	3.1	0.00
5	487	119.7	10.9	0.02	*15*	309	125.9	35.5	0.01
6	452	109.8	14.4	0.02	*16*	162	95.5	11.1	0.02
7	432	441.4	44.8	0.03	*17*	162	115.8	15.5	0.05
8	392	136.7	19.9	0.02	*18*	675	148.5	24.0	0.02
9	282	320.7	48.7	0.02	*19*	339	88.4	13.3	0.07
10	251	102.3	8.7	0.02					

MC (A-PUA)	Size	Average Length	SDL	LC fraction	MC (A-PUA)	Size	Average Length	SDL	LC fraction
A1	69369	223.7	29.9	0.02	*A28*	1365	83.9	13.2	0.02
A2	8908	203.9	19.1	0.05	*A29*	615	84.5	8.4	0.04
A3	8324	49.1	5.6	0.00	*A30*	506	47.7	4.8	0.01
A4	4523	158.6	36.5	0.02	*A31*	464	99.5	10.8	0.01
A5	3559	210.2	28.7	0.03	*A32*	406	191.7	18.6	0.06
A6	3386	193.1	17.9	0.05	*A33*	340	183.3	33.4	0.02
A7	2934	102.7	12.1	0.02	*A34*	294	99.0	10.6	0.03
A8	2915	347.2	*76.5	0.04	*A35*	285	48.8	6.6	0.01
A9	2870	392.0	48.9	0.06	*A36*	248	59.6	7.5	0.03
A10	2735	257.3	12.6	0.02	*A37*	198	210.5	30.9	0.03
A11	2392	146.5	40.2	0.03	*A38*	1588	226.5	45.2	0.01
A12	1795	153.3	11.4	0.01	*A39*	691	86.7	19.7	0.00
A13	1751	235.9	25.9	0.03	*A40*	565	369.6	35.2	0.04
A14	986	289.8	*57.4	0.05	*A41*	430	339.6	36.9	0.01
A15	851	164.1	13.9	0.03	*A42*	359	328.7	*52.0	0.03
A16	839	193.3	26.1	0.01	*A43*	267	165.1	32.6	0.08
A17	700	259.4	29.3	0.03	*A44*	208	36.3	3.6	0.00
A18	556	46.8	5.0	0.02	*A45*	186	181.8	37.9	0.09
A19	452	173.0	17.0	0.02	*A46*	121	311.9	26.4	0.04
A20	384	189.3	32.3	0.01	*A47*	110	211.3	23.1	0.02
A21	193	293.9	36.0	0.02	*A48*	677	87.5	15.4	0.02
A22	190	114.7	13.7	0.02	*A49*	625	119.5	19.9	0.03
A23	172	43.0	4.2	0.00	*A50*	277	77.9	6.4	0.01
A24	162	60.4	4.8	0.01	*A51*	178	132.0	15.4	0.01
A25	146	86.6	16.1	0.04	*A52*	126	62.2	10.2	0.11
A26	135	216.7	12.3	0.02					
A27	3181	196.9	21.1	0.02					

Table 3.3: Member region's statistics for PUA_UR50 MCs. Top section: MCs containing PUA domains; bottom section, MCs containing PUA-associated domains (A-PUA, with ”A” prefix). For each MC, we report size (i.e., number of sequence members), average and standard deviation of members' lengths and, the fraction of residues (of all members) that are found in low-complexity regions (LC fraction). We flag MCs (*) for which the SDL is larger than 50 amino acids (about the size of a small domain).

3.2.2.2 Comparison between MCs and Pfam families boundaries

Another important aspect of comparing two protein classifications entails investigating by how much the boundaries of the respective clusters or families differ when evaluated on the same sequences. The quantities F_{MC}^{ext} and F_{MC}^{red} in Table 3.1 indicate the extent of the agreement between the boundaries of DA members and the respective Pfam annotations, as explained in Section 3.1.3 .

Equivalent MCs are the closest to the DA architectures in terms of their boundaries; the other categories feature cases that may be worthy of further inspection. MC-A29_PUA, for example, features member regions that typically cover only about half of the DA family tRNA (Uracil-5-)-methyltransferase (PF05958) as annotated by Pfam in the full-length proteins they belong to. Structural data indicate that, in fact, Pfam family PF05958 covers two structural domains: a so-called central domain, which hosts a [Fe4S4] cluster, and a catalytic domain typical of SAM-dependent methyltransferases. MC-A29_PUA covers only the catalytic domain of the tRNA (Uracil-5-)-methyltransferase, albeit imperfectly (see Figure 3.12). In another example, the MC-18_PUA($F_{ext}^{MC} = 0.47$) metacluster we already discussed, the 'true' DA is likely to be constituted of the double-domain architecture pre-PUA+PUA rather than by PUA only.

Interesting cases are constituted by MC-1_PUA and MC-17_PUA, both mapping to the Lon_substr_Bdg family (PF02190). While the first MC has "equivalent" status, the second one is a "reduced" MC mapping only to half of the domain. However, the Lon_Substr_bdg domain contains two structural units (see Supp. Fig. S5), of which MC-17_PUA captures only the first.

When discussing boundaries, we should not forget that some MCs feature a bi-modal distribution of their members' lengths (see for example Figure 3.8 A and B). In these cases, the average measures F_{ext} and F_{red} cannot capture the full complexity of boundary differences with respect to the Pfam annotation.

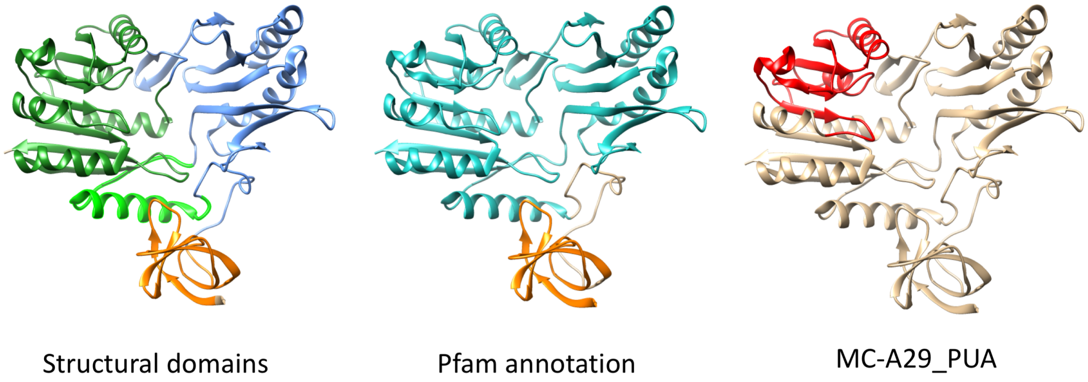


Figure 3.12: X-ray crystal structure of RumA, an E.coli class I SAM-dependent methyltransferase (PDB:2bh2_A). Structural domains (following (Lee et al., 2004): N-terminal domain (orange, aa15-74), Central domain (light blue, aa75–92 and 125–262) and C-terminal (catalytic) domain (light green, aa93–124 and green, aa263–431); (center) Pfam annotation: TRAM (PF01938) (orange, aa10-67) and tRNA_U5-meth_tr (PF05958) (cyan, aa95-432); (right) Region that aligns (HMMER online v3.3) to the profile-HMM built of MC-A29_PUA (aa285-369, red).

3.2.2.3 MCs with minimal Pfam annotation

Three MCs with single-family DA feature low %DAC and high %DACF indicating that, for the most part, they are constituted of member sequences that are devoid of any Pfam annotation; these are MC-A7_PUA, MC-A39_PUA and MC-A48_PUA. MC-A48_PUA member regions, 92% of which are unannotated, are found in ATP-dependent Lon protease proteins and typically cover a helical region located at the N- terminus of the AAA (PF00004) ATPase domain (Figure 3.13). This region could potentially be built into a short “pre-AAA” motif. MC-A7_PUA (54% of unannotated members) and MC-A39_PUA (46.5%) map, respectively, to tetratricopeptide-like repeats or TPRs (CL0020) and Cys3His zinc-binding domains (CL0537) also often found in tandem repeats. Tandem repeats such as these, which are relatively short and often feature a high degree of divergence in sequence, are notoriously difficult to classify exhaustively. It is thus not surprising that many elements of these MCs do not carry annotation in Pfam. Note that increases of %DACFA in these two MCs are mostly due to the presence of members with a higher number of repeated domains than found in the DA. There might be scope for using these MCs as a basis to boost coverage of the respective clans.

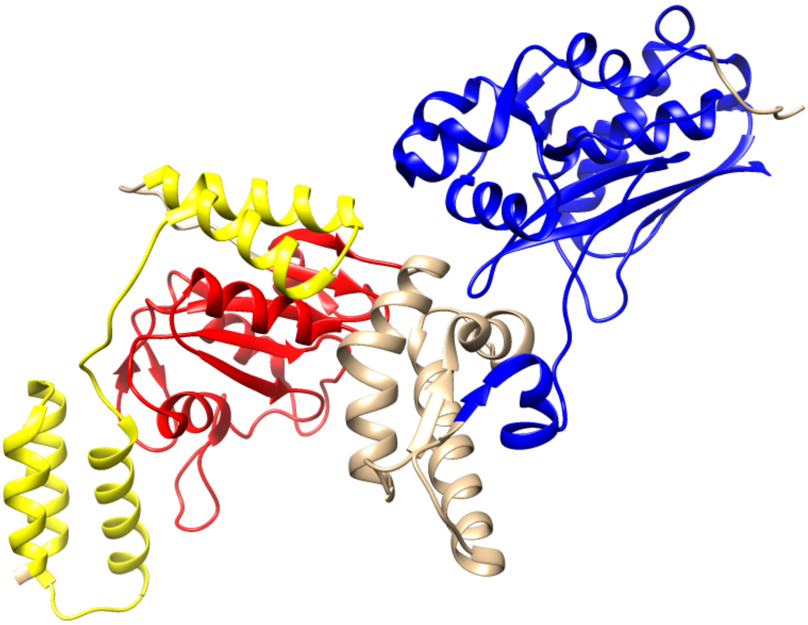


Figure 3.13: Structure of PDB protein chain 4ypl_A (protein A0A059VAZ3). Red and blue section show Pfam annotation: AAA region in red (aa. 351-491) and LON_C in blue (aa. 568-772). Yellow region (aa. 245-339) shows the hit of MC-A48_PUA profile-HMM.

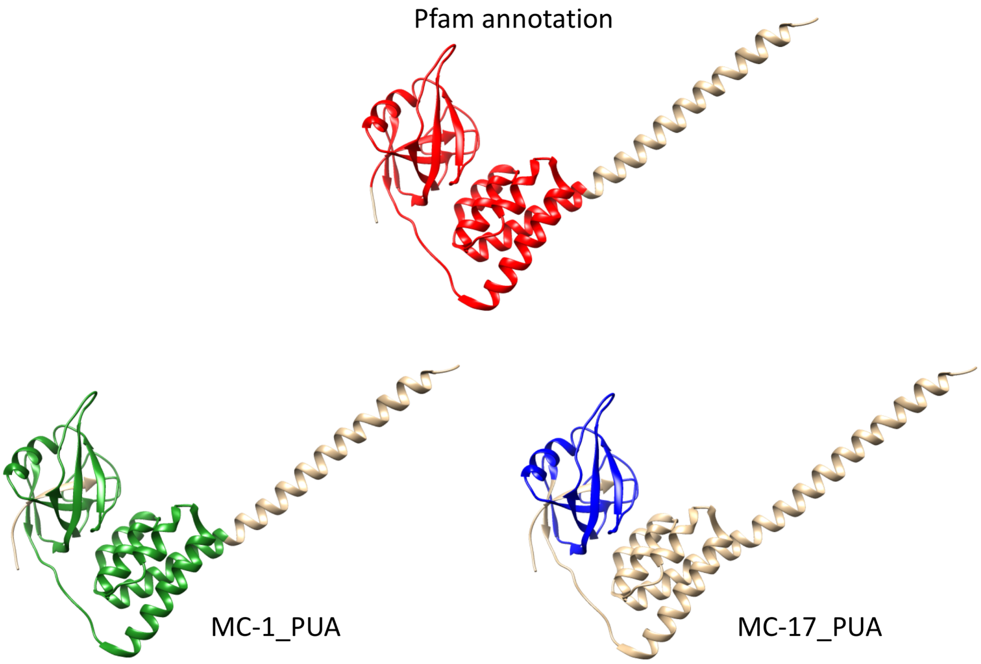


Figure 3.14: Structure of PDB protein chain 3ljc_A (protein P0A9M0). Red section show Pfam annotation (Lon_substr_bdg aa. 10-202). Green region (aa. 17-207) shows the hit of MC-1_PUA profile-HMM (E-value 2.2E-26); blue region (aa. 17-109) shows the hit of MC-17_PUA profile-HMM (E-value 1.1E-5).

3.2.2.4 Degeneracy of MCs with respect to Pfam families

In some instances, DPCfam produces multiple clusters that map to the same Pfam family or group of families. Here it is worth pointing out that we cluster alignments rather than protein sequences. This means that alignments of the same protein region to different proteins are treated as separate entities. DPCfam tries to ensures that when two regions of the same protein of about the same size have a large overlap, they are classified as belonging to the same cluster. For overlaps that are small with respect to the length of the alignments being compared, the regions may end up in different MCs. One such example is represented by the trio of clusters MC-A30_PUA, MC-A13_PUA and MC-A49_PUA, all of which feature the same DA, namely, TruB_N + TruB_C_2 (PF01509+PF16198). Both of these Pfam families are part of the PseudoU_synth (CL0649) clan. In Figure 3.15, taking as sample sequence one for which a structure is available, we show that although the 3 clusters share the same DA, the actual set of families they cover is quite different. In fact, the three MCs belongs to three different boundary categories (see Table 3.1): reduced (MC-A30_PUA), equivalent (MC-A13_PUA) and shifted (MC-

A49_PUA). Contrary to MC-A13_PUA, that truly corresponds to the DA families, MC-A30_PUA covers mainly TruB_C_2 with minimum overlap to the first family, and MC-A49_PUA covers mainly TruB_C_2, but also extends beyond it in a region that when annotated is reported to be part of a PUA-clan family.

In the Pfam clan, the pseudouridine synthase domain has sometimes been split into two families (TruB_N +TruB_C_2, PseudoU_synth_1x2, PseudoU_synth_1+DUF2344) or otherwise classified as a single family (PseudoU_synth_2, TruD). The difficulty for a consistent evolutionary classification of this domain comes primarily from two things: (i) the pseudouridine synthase domain appears to be formed by a tandem duplication the two moieties of which share often very little sequence similarity with each other (and only structural similarity in terms of their general topology) and (ii) the two homologous moieties feature strand swapping and sometimes nesting of additional domains. The latter is the case for sequences in the TruD family, which in Pfam additionally covers a nested domain that should instead be built as a separate family outside of the CL0649 clan (see Figure 3.16). Also, the boundaries of paired families such as TruB_N and TruB_C_2 do not seem to reflect the structural organization of the duplication very well (see red and blue regions in Figure 3.17). Indeed, the current boundaries of the two families represent regions of very different structure, with the TruB_C_2 open and elongated structure not reminiscent of a typical structured domain. There is, for example, no pairwise structural alignment produced by DALI with default settings for the TruB_N and TruB_C_2 Pfam annotated regions of PDB structure 3u28 A. We suggest that building a family covering the entire pseudouridine synthase domain would also in this case (as in, for example, PseudoU_synth_2) be the best option. Finally, the Pfam nomenclature of families that map to tRNApseudouridine synthase B proteins is quite confusing. TruB_N is the N-terminal part of a PseudoU_synth domain, TruB_C is a PUA domain, TruB_C_2 is the C-terminal part of a PseudoU_synth domain and TruB-C_2 is again a PUA domain. Although we understand family names have a historical relevance, a rethinking of this particular set of names may be beneficial. It is interesting to note that in our automatic classification the N-terminal boundary of the TruB_C_2 family is well matched by both MC-A30_PUA and MC-A49_PUA, highlighting the distant evolutionary relationship between the two moieties of the pseudouridine synthase domain.

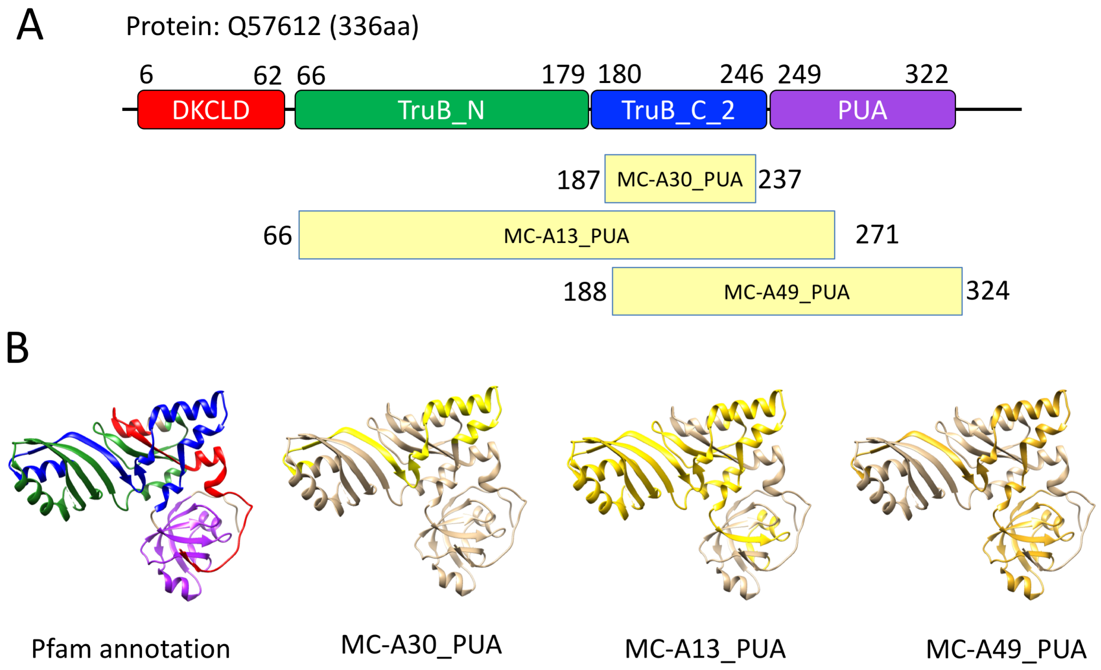
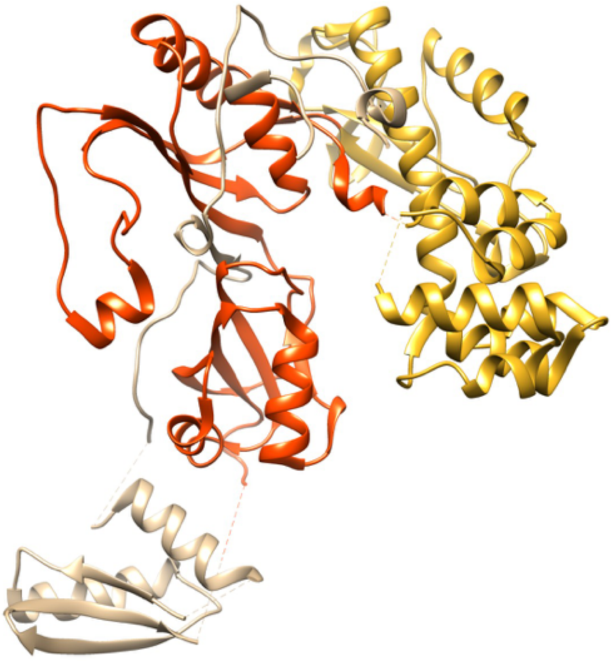


Figure 3.15: Annotation for protein Q57612 (2apo in pdb). (A) Sequence with Pfam annotation; yellow boxes shows the hits obtained using profile-HMMs derived from the metaclusters MC-A30_PUA, MC-A13_PUA and MCA-49_PUA. (B) Structure of 2apo PDB chain A, colored following Pfam classification in panel A and according to the matches with the profile-HMMs of MC-A30_PUA (aa 187-237) in yellow, MC-A13_PUA (aa 66-271) in gold and MC-A49_PUA (aa 188-324) in dark gold.

3.2.2.5 Coverage of the PUA clan by DPC-generated MCs.

So far, we have looked at how consistent the Pfam annotations are within the DPCfam MCs (in other words, we looked at the accuracy of our classification). Clearly, it is also important to know to what extent the automatically-generated classification recapitulates Pfam's coverage of the sequence space. Here we investigate coverage of PUA clan regions within the UniRef50 database: we consider all regions that produce significant alignments to the MCs-derived profile-HMMs (hmmsearch run against PUA_UR50, sequence E-value <0.01, Hit E-value <0.03). We say that an MC covers a PUA region if there is at least one of the profile-HMM hits covers >=75% or =100% of it. We plot the cumulative coverage of the Pfam PUA clan when ranking MCs from the one that contributes the highest coverage to the one that contributes the lowest coverage (Figure 3.18); we note that proteins are counted only once, even if covered by more than one MC. It is interesting to see that coverage converges after a number of MCs that is roughly equivalent to



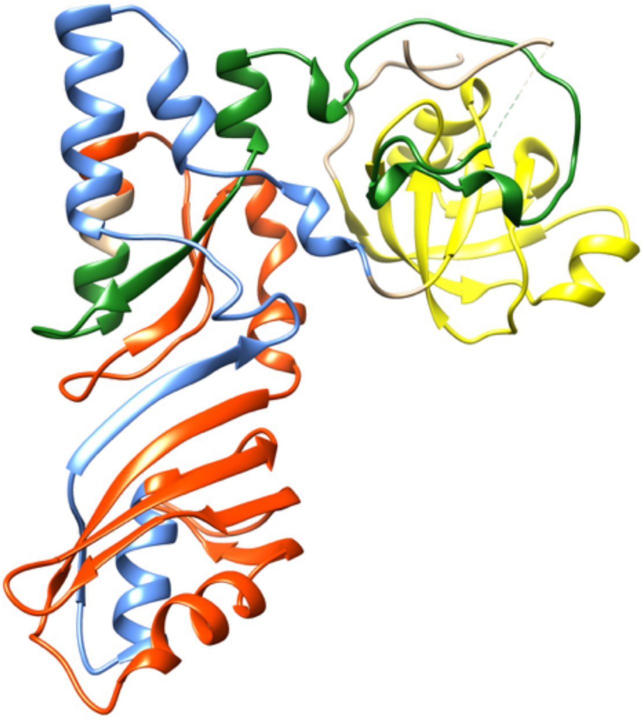
.

Figure 3.16: Structure of PDB protein chain 5kkp_A. The colored region (red+gold) covers a TruD domain as annotated in Pfam. The nested gold domain (roughly, aa384-577) is not related in structure to domains in the PseudoU_synth clan and, as such, should be built as a separate Pfam family not part of the clan. Regions not annotated in Pfam are colored tan. TruD annotation according to Pfam 32.0.

the number of Pfam families in the PUA clan (11 total). We see that >80% of the PUA clan regions are covered for at least 75% of their length by the top 15 clusters. While a fraction of Pfam regions in the PUA clan is not covered by MCs, we should point out that most PUA-covering MCs include at least some additional regions not currently annotated in Pfam, which are likely to represent new clan members. The flattening out of the curves that we observe after 10-15 MCs reflects the fact that most of our 71 MCs cover PUA-associated families rather than PUA families.

3.2.3 Clustering of proteins from the P53-like clan

Clustering of the PUA clan, which is described in detail in the previous sections, has uncovered several interesting features of the relationships between the Pfam families involved. Our clustering procedure utilizes few adjustable parameters and we did not perform any systematic exploration of the parameter space. Rather,



.

Figure 3.17: Structure of PDB protein chain 3u28_A. Colors identify Pfam families annotated on the structure (from N- to C-terminus): DKCLD (green), TruB_N (red), TruB_C_2 (blue) and PUA (yellow). Regions not annotated in Pfam are colored tan. Annotation according to Pfam 32.0.

parameters were mostly chosen following heuristic rules from the literature, thus considerably limiting the risk of over-fitting. Nevertheless, we did use the PUA clan to tune some aspects of our procedure (e.g. thresholds for merging MCs). As a consequence, in this section, we report on results obtained when running DPC-based clustering on a second Pfam clan, when all parameters have been left unchanged with respect to the ones used for PUA. This should provide additional evidence of the fact that our method could be successfully extended to the analysis of larger portion of the sequence space. In particular, we run our DPC procedure on the P53-like clan (Tables 3.4 and 3.5). Overall, the results appear to be in line with the ones obtained for PUA. Our procedure generates 28 MCs of size >100, of which 53.6% have %DAC> 95%. Only two MCs, MC-19_P53 and MC-28_P53, have %DACFA< 98%. MC-19_P53 is peculiar in that the vast majority of its members lack Pfam annotation (+95.4% in the %DACF column with respect to the single-domain DA). This may be explained by the high value of the low-complexity residue fraction in

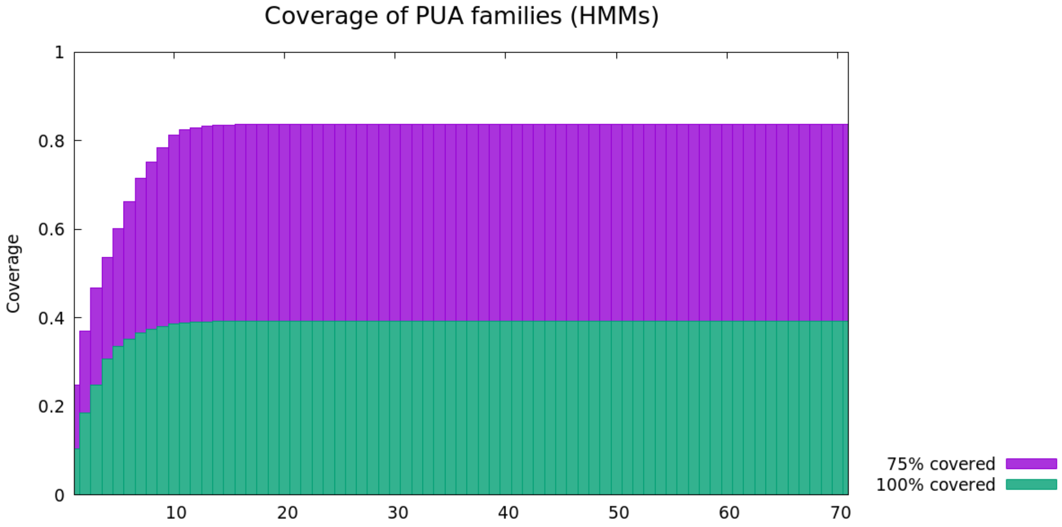


Figure 3.18: Coverage of proteins in UniRef50 including at least a region of the PUA clan according to Pfam-A. By running the profile-HMMs derived from the metaclusters, we search for hits with at least 75% of coverage and 100% of coverage. The graph shows the fraction of the PUA clan covered using an increasing number of metaclusters. After after the 15-th metacluster, the fraction does not improve, because of some redundancy in PUA MCS, which are 19. Also, MCs appearing after the 19-th do not map to a PUA clan family, and do not contribute to any increase in the coverage.

this MC ($LC = 0.58$, Table 3.4), suggesting that its member regions are unlikely to represent a structural domain. Additionally, low-complexity regions are more likely to align to non-homologous sequences (thus %DACFA=95.9%). MC-28_P53 contains 132 sequences, 54% of which are not annotated, 33% annotated as PF09270 (BTD), 7% annotated as PF01833 (TIG) and, finally, 5% annotated as BTD + TIG. BTD is not a P53-like family, however, it is found by our clustering algorithm because BTD is commonly located at the C-terminus of the P53-like LAG1-DNAbind family. Although the BTD annotation is the most present in MC-28_P53, the domain it represents is poorly covered. Indeed, only a few amino-acids at the C-terminus of BTD are found in MC-28_P53 members. On the contrary, when present, TIG regions are well covered. Searching the Reference Proteome dataset with a MC-28_P53 generated profile-HMM we found 2,083 significant hits (hmmsearch, sequence E-value <0.01, Hit E-value <0.03). About half of these mapped to TIG domains, while the rest although often found C-terminal to a LAG1-DNAbind + BTD architecture are not annotated in Pfam. Finally, we ran MC_28-P53 profile-HMM against the PDB, finding as top matches yet unannotated regions located at the C-terminus of LAG1-DNAbind + BTD architectures (see Figure 3.21 A-C for an example). Of these, we focused on SUH_HUMAN (Q06330, PDBid 3nbn_A). The region of 3nbn A aligned to the MC-28_P53's profile-HMM appears to be well-structured (Figure 3.21 B, yellow) and it is structurally similar to TIG domains (Figure 3.21 C.). In conclusion, MC-28_P53 is likely to represent a TIG family covering a good number of TIG domains not yet annotated in Pfam. Coverage of Pfam P53-like clan's regions by P53 MCs is comparable to the one observed for the PUA clan (see Figure 3.19).

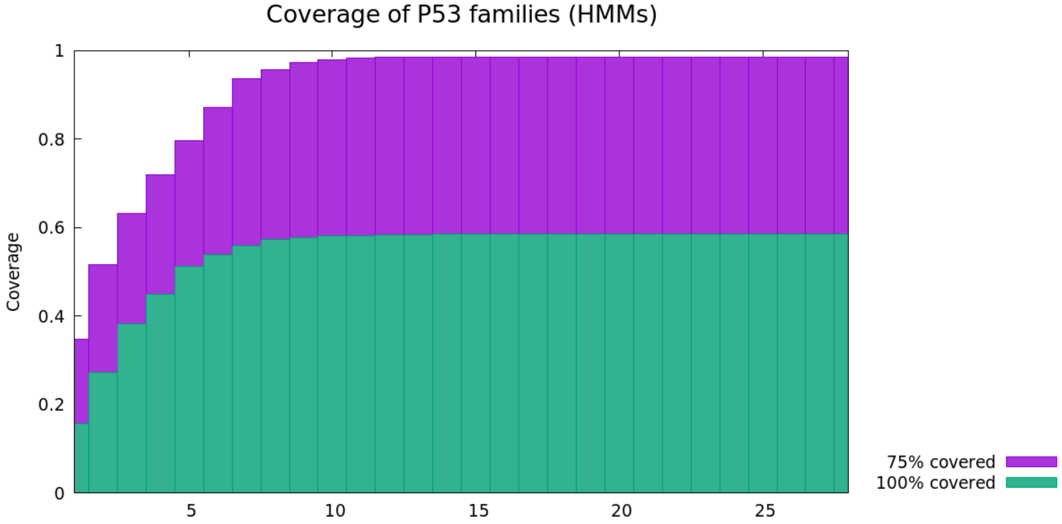


Figure 3.19: Coverage of proteins in UniRef50 including at least a region of the P53-like clan according to Pfam-A. By running the profile-HMMs derived from the metaclusters, we search for hits with at least 75% of coverage and 100% of coverage. The graph shows the fraction of the P53 clan covered using an increasing number of metaclusters.

In general, in the case of the P53 clan, we notice two main differences with respect to the clustering of the PUA clan. First, we see what appears to be a higher degree of MC redundancy with respect to the Pfam classification. For example, 6 MCs have PF00907 as their DA and 4 MCs feature PF05224 in theirs. It should be noted, however, that in the case of PF00907 only two MCs have an average length of more than 50aa. In fact, MC-14_P53 and MC-27_P53 have length <30aa, which is shorter than the length of the average protein domain [56]. In Figure 3.20 we show a graphical view of how the different MCs map to this Pfam family. Second, with respect to the PUA clan, on average, MC boundaries appear to match less well those of the DA families. Indeed, in Table 3.4 we observe several MCs with high F_{ext}^{MC} and/or F_{red}^{MC}. We notice, again, that this is often the case for MCs of short average length.

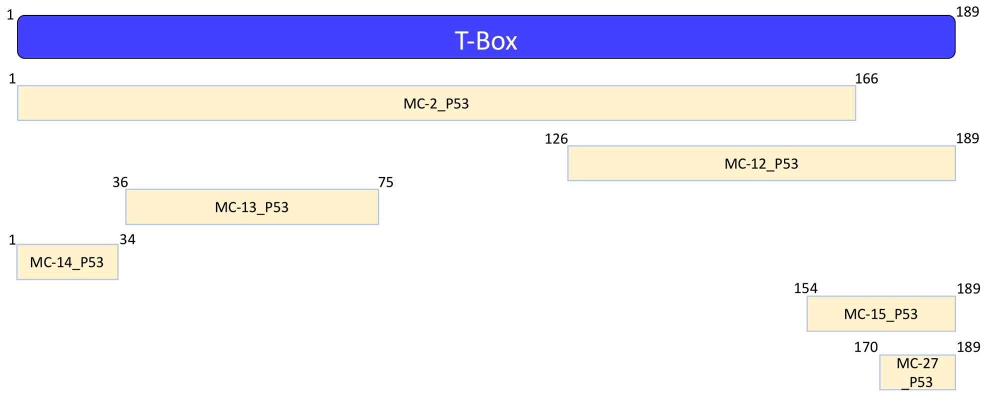


Figure 3.20: Coverage of P53_UR50 redundant metaclusters with respect to their common PF00907 (T-Box) DA. We used HHpred to determine the position of each MC with respect to the T-box profile-HMM (the first match for all these MCs, with hhpred probability>98%).

MC	Size	Average Length	SDL	LC fraction	MC	Size	Average Length	SDL	LC fraction
1	25859	185.9	30.8	0.02	*16*	281	126.1	13.5	0.01
2	941	171.9	26.9	0.01	*17*	254	68.2	7.4	0.01
3	481	279.4	30.7	0.01	*18*	213	45.8	2.9	0.00
4	467	465.9	*92.3	0.01	*19*	194	111.6	15.0	0.58
5	462	204.5	31.0	0.02	*20*	154	39.0	3.1	0.02
6	231	494.3	*106.3	0.02	*21*	145	34.0	3.5	0.01
7	231	340.8	*52.0	0.05	*22*	9428	207.9	17.6	0.03
8	225	191.1	29.7	0.02	*23*	699	135.1	21.9	0.02
9	166	475.7	*107.6	0.02	*24*	124	399.5	*58.8	0.03
10	163	126.0	10.4	0.01	*25*	203	136.3	15.9	0.01
11	110	421.4	*53.4	0.05	*26*	158	111.1	14.2	0.01
12	761	67.8	8.7	0.01	*27*	137	28.2	2.6	0.00
13	531	43.6	4.4	0.00	*28*	132	137.7	21.0	0.05
14	525	29.6	2.7	0.00					
15	363	38.2	5.0	0.00					

Table 3.4: Member region's statistics for P53_UR50 MCs (see Table 3.3).

MC	DA	% DAF	%DAC	%DACF	%DACFA	F_{ext}^{MC}	F_{red}^{MC}	Extra Clans
1	PF12796_PF12796	26.1	52.2	68.0	100.0	0.10	0.11	
2	PF00907	97.7	97.7	97.7	100.0	0.03	0.02	
3	PF00554_PF16179	88.8	88.8	96.7	100.0	0.03	0.00	
4	PF00400_[...]_PF00400	18.2	19.3	52.9	100.0	0.02	0.04	
5	PF05224	98.7	98.7	99.8	100.0	0.16	0.01	
6	PF02865_PF01017_PF02864_PF00017	29.0	29.0	95.2	100.0	0.06	0.11	
7	PF09271_PF09270	86.6	86.6	92.6	100.0	0.13	0.01	
8	PF00870	84.9	84.9	88.0	100.0	0.07	0.06	1
9	PF03068	53.0	53.0	53.0	100.0	0.12	0.05	2
10	PF00853	99.4	99.4	99.4	100.0	0.04	0.04	
11	PF09751	88.2	88.2	88.2	100.0	0.16	0.01	
12	PF00907	98.3	98.3	99.1	100.0	0.03	0.57	
13	PF00907	99.2	99.2	100.0	100.0	0.01	0.73	
14	PF00907	98.1	98.1	100.0	100.0	0.13	0.76	
15	PF00907	99.4	99.4	99.7	100.0	0.14	0.77	
16	PF00554	99.6	99.6	99.6	100.0	0.03	0.28	
17	PF05224	99.6	99.6	100.0	100.0	0.04	0.61	
18	PF00554	100.0	100.0	100.0	100.0	0.08	0.72	
19	PF15709	0.5	0.5	95.9	95.9	0.11	0.41	
20	PF00853	100.0	100.0	100.0	100.0	0.06	0.60	
21	PF09271	97.9	97.9	100.0	100.0	0.07	0.69	
22	PF00005	95.8	95.8	95.8	100.0	0.30	0.02	
23	PF13884	82.5	82.8	83.5	100.0	0.57	0.00	
24	PF05224_PF13884_PF13887	85.5	85.5	98.4	100.0	0.24	0.00	
25	PF02864_PF00017	67.5	67.5	99.5	99.5	0.21	0.60	
26	PF05224	98.7	98.7	100.0	100.0	0.27	0.57	
27	PF00907	98.5	98.5	99.3	100.0	0.38	0.87	
28	PF09270	33.3	33.3	87.9	93.2	0.94	0.93	1

Table 3.5: DA annotation of P53_UR50 MCs (see Tables 3.1 for column description). Highlighted in bold MCs contain P53 domains. DA including "[...]" represent a very long repeat, which has not been reported entirely for formatting reasons.)

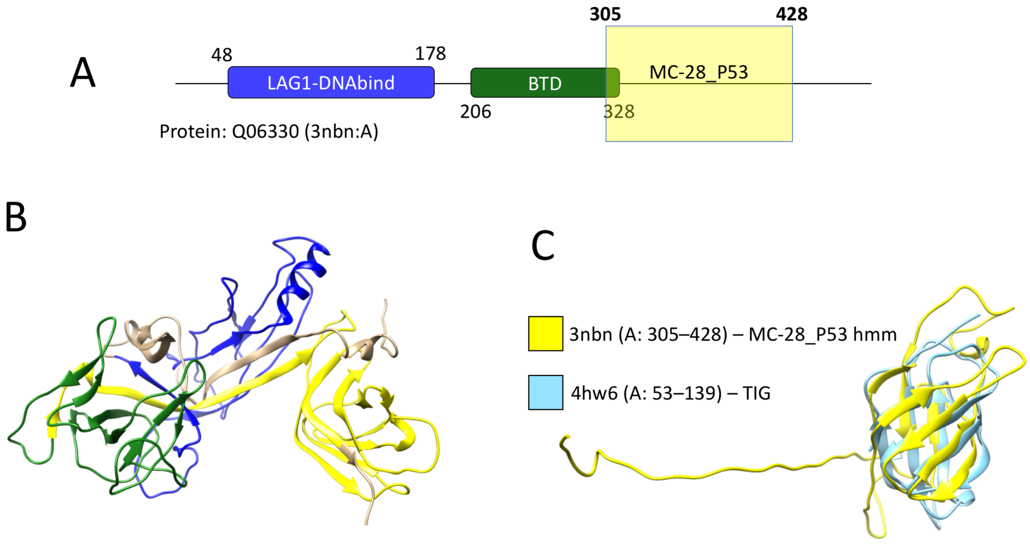


Figure 3.21: Annotation for protein Q06330 (A): Sequence Pfam annotation; the yellow box marks a hit obtained using the profile-HMM derived from metacluster MC-28_P53. (B): Pfam and MC-28_P53 annotations of panel (A) mapped to one of the available structures of Q06330 (PDBid 3nbn:A). Color code for families and regions is the same as in (A). (C): Structural alignment between the MC-28_P53's annotated region of 3nbn (yellow) and the TIG domain of PDB structure 4hw6 (light blue). Alignment obtained with DALI pairwise online tool; alignment features: Z=6.2, RMSD=2.2, percent sequence identity=25).

3.2.4 A robustness test for DPCfam parameters

We already mentioned that our clustering procedure utilizes few adjustable parameters, mostly chosen following heuristic rules from the literature: this means that no systematic exploration of the parameters' space has been done. We note that such a procedure would have required to define some quality measure on MCs with respect to a ground truth. In the previous sections showed how the concept of Ground Truth in this case is complicated to define and to relate with, and that we may not dislike finding MCs in disagreement with the Pfam classification, since in Pfam it is possible to find unannotations or boundaries imperfections. Nonetheless, we still wanted to test the robustness of the parameters used, to be sure that the DPCfam results are stable with respect to small variations on its main parameters, namely μ_1, μ_2 and Δ. Thus, we tested (*a posteriori*) the robustness of DPCfam metaclustering of P53_UR50, with respect to small variations (±10%) of μ_1, μ_2 and

Δ. In addition, we tested the case of reducing by half the size of the query sequence dataset. In particular, we compare the assignment of alignments to metaclusters before the filtering step (see Table 3.6). In our comparison, we use: i) the number of alignments that are assigned to metaclusters; ii) the percenage of alignments metaclustered with the standard set of parameters that are still assigned to metaclusters when utilising the modified parameters; iii) the Normalized Mutual Information. The NMI is given by $\mathrm{NMI}(C_1, C_2) = \frac{2I(C_1,C_2)}{H(C_1)+H(C_2))} \in [0,1]$, where C_1 and C_2 are the class labels assigned to alignments by two different clustering procedures, I is the Mutual Information between the two classifications and $H(C)$ is the entropy of a single classification. Two identical classifications gives NMI$= 1$. To compute NMI, we consider those alignments that have been metaclustered by both the reference and the alternative clustering procedure (i.e, those counted in the third column of Table 3.6).

Parameters	**Number of alignments metaclustered**	**Percentage of $\mu_1 = 0.2$, $\mu_2 = 0.9$, $\Delta = 0.5$ alignments' metaclustered**	**NMI over common alignments**
$\mu_1 = 0.2$, $\mu_2 = 0.9$, $\Delta = 0.5$	1,350,496	-	-
$\mu_1 = 0.2 + 10\%$	1,484,231	94%	0.96
$\mu_1 = 0.2 - 10\%$	1,351,613	89%	0.99
$\mu_2 = 0.9 + 10\%$	1,494,990	95%	0.99
$\mu_2 = 0.9 - 10\%$	1,259,045	93%	0.99
$\Delta = 0.5 + 10\%$	1,326,511	98%	0.99
$\Delta = 0.5 - 10\%$	1,374,507	98%	0.95
Query Dataset	**Number of 50% P53_UR50 alignments metaclustered**	**Percentage of 644,648 alignments metaclustered**	**NMI over common alignments**
P53_UR50	644,648	-	-
50%_P53_UR50	642,223	99%	0.99

Table 3.6: *Metaclusters' robustness upon ±10% variation of the μ_1, μ_2, Δ parameters and, additionally, when reducing by half the number of query sequences.* Test are performed on the P53_UR50 dataset. We consider alignments assigned to metaclusters before the filtering step (see Methods). $\mu_1 = 0.2$, $\mu_2 = 0.9$, $\Delta = 0.5$ are the parameters used throughout the main analysis. NMI stands for Normalized Mutual Information (see Results for the definition).

In general, parameters' variation does not result in significant changes in the number of alignments assigned to a metacluster. Variations in μ_1 and μ_2 imply smaller or larger cutoffs in the density estimations, and a more or less restrictive criterion for assigning alignments to cluster centers. Not surprisingly, larger values of μ_1 and μ_2 produce more metaclustered alignments, while smaller values produce less (see second column of Table 3.6). This is reflected also in the percentage of alignments metaclustered using the standard procedure that are also retrieved when varying μ_1 and μ_2, with smaller percentages obtained using smaller values (see third column of Table 3.6). However, despite of these differences, the NMI with the results obtained with the reference setup is always extremely high, indicating that the results are robust with respect to the choice of this parameter. Different values of Δ result in adding or removing density peaks: the small variations performed do not change significantly the number of alignments metaclustered (changes of about 2%), covering the vast majority (98%) of the alignments clustered with the standard procedure. Also in this case the NMI with respect to the reference setup is very high. We also repeated the whole procedure on a query dataset containing only half of the sequences, selected at random (50% P53_UR50). In this analysis we collect 642,223 alignments in metaclusters. In the same subset of sequences, performing the analysis on the full dataset we assign to metaclusters 644,648 alignments. Almost all these alignments are in common (see third column of Table 3.6) and, consistently, the NMI between the two metacluster partitions is 0.99.

3.2.5 A discussion on the proof-of-principle results

In this proof-of-principle experiment we showed that, in most cases, automatically-generated metaclusters (MCs) represent single or multi-domain architectures which, overall, display a good agreement with the Pfam annotation. With respect to the presence of multi-domain MCs, we emphasize that our procedure clusters evolutionary modules (using sequence similarity) rather than directly structural domains: it may be difficult for our method to split into separate MCs structural domains that are only (or overwhelmingly) observed in joint architectures, unless these domains are separated by long regions of low conservation. We also observe, especially in the analysis of the P53-like clan, a certain degree of redundancy between MCs (i.e. multiple MCs mapping to the same Pfam family). Although it is possible that this redundancy could be significantly reduced by discarding short length MCs, we think that the results indicate that MC-merging step of DPCfam could potentially be improved.

In general, significant differences between clans exist in terms of size, evolutionary divergence, complexity of architecture and structural class of their families. Although these diversity cannot be recapitulated in full by the analysis of only two

Pfam clans done in this section, it is worth pointing out that our clustering experiment allowed finding numerous families outside of the PUA and P53-like clans (see Tables 3.2 and 3.5). This is due to the fact that our method runs on full-length sequences and that about 45% and 39% of PUA and P53-like member regions, respectively, are part of multi-domain proteins.

These preliminary result brought us to believe that DPCfam can support manual annotation by pointing to opportunities for expanding existing families or clans and, occasionally, by identifying inconsistencies in the current classification (including incorrect domain boundaries and incomplete clan memberships). Given these encouraging results, we decided to optimize the computational implementation, as discussed in Section 2.4, and make it suitable to perform the full clustering of UniRef50.

3.3 All-to-all clustering of UniRef50

In this section we will present the results obtained from the all-to-all clustering of UniRef50, performed as described in Section 2.4. The analysis is performed, we recall, on UniRef50 v. 2017_07, which counts 23,531,980 protein sequences.

First, we will show some results with measures performed without making reference to a ground truth; then we will compare our metaclusters to Pfam. Due to the quantity of MCs to analyse, we avoided any kind of deep, protein-wise analysis of the MC content, as done in Section 3.2. We instead used general measures based on what we learned from the proof-of-concept analysis. Being a large scale analysis, a direct comparison with Pfam becomes meaningful: to this extent, we focused more on coverage measures, and we will dedicate an entire subsection to the subject.

DPCfam produced 210,802 metaclusters. Most of analysis presented here will consider MCs and their content *before* applying 95 P.I. reduction with Cd-Hit.

3.3.1 General properties of DPCfam MCs

The 210,802 MCs generated display a power-law distribution in sizes (Figure 3.22), namely in the number of protein regions they contain. This is an encouraging result, considering that it has been observed that protein families display power-law like distribution occurrences [57]. This distribution is normally ascribed to the biological processes at the base of evolution, such as gene duplication or gene generation, driving the protein families' generative process.

Figure 3.23 show the MCs' regions average length, here named AL, with the respective standard deviation (SDL) as error-bar. MCs are sorted with respect to AL. From panel A, we see that most of the MCs have AL> 50 (88% of MCs), and that is not frequent to find MCs with AL> 500 (4% of MCs) . Panel B shows the ratio SDL/AL, which is distributed around 0.1 and rarely larger than 0.2 (5.5% of MCs). This suggests that regions collected in a MC are consistent in terms of amino-acidic lengths.

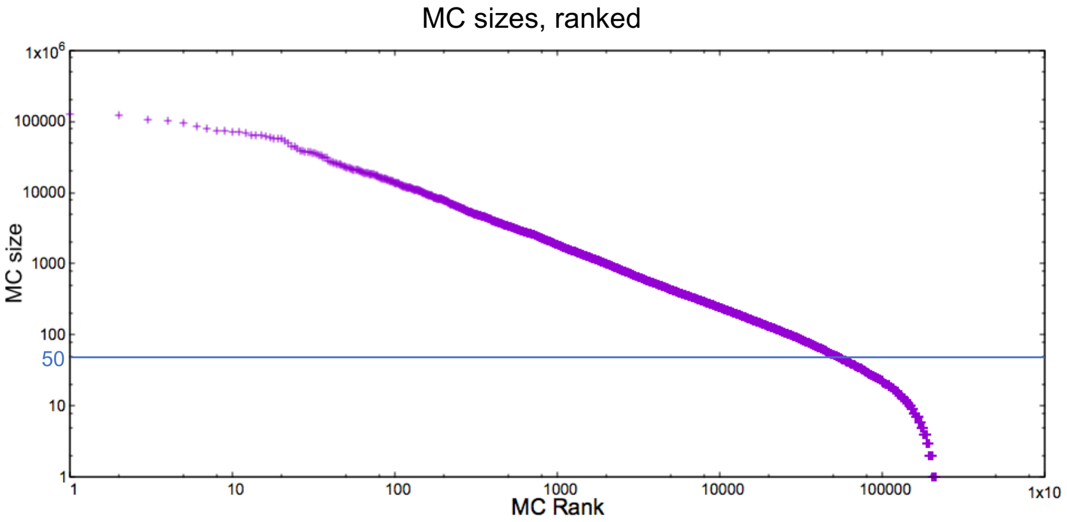


Figure 3.22: Metacluster's sizes with respect to their rank. MCs' are sorted with respect their size (number of sequences), obtaining this rank-size plot, in a log-log scale. We note that this kind of plot is strictly related to the cumulative distribution of MC sizes, and it shows a strong power-law behaviour.

From these considerations, we decided to include in the downstream analysis only those MCs that have more than 50 regions and whose regions' average length is larger than 50 a.a., which results in 45,679 MCs.

We next computed the Low Complexity (LC) fraction for each MC: this is the fraction of amino-acids found in a low complexity regions of the MC over the total number of amino-acid in the MC. The average value of LC fraction of all MCs is 0.04, with 4,743 MCs (10%) with an LC fraction larger than 0.1 and only 698 MCs (1,5%) with an LC fraction larger than 0.2 .

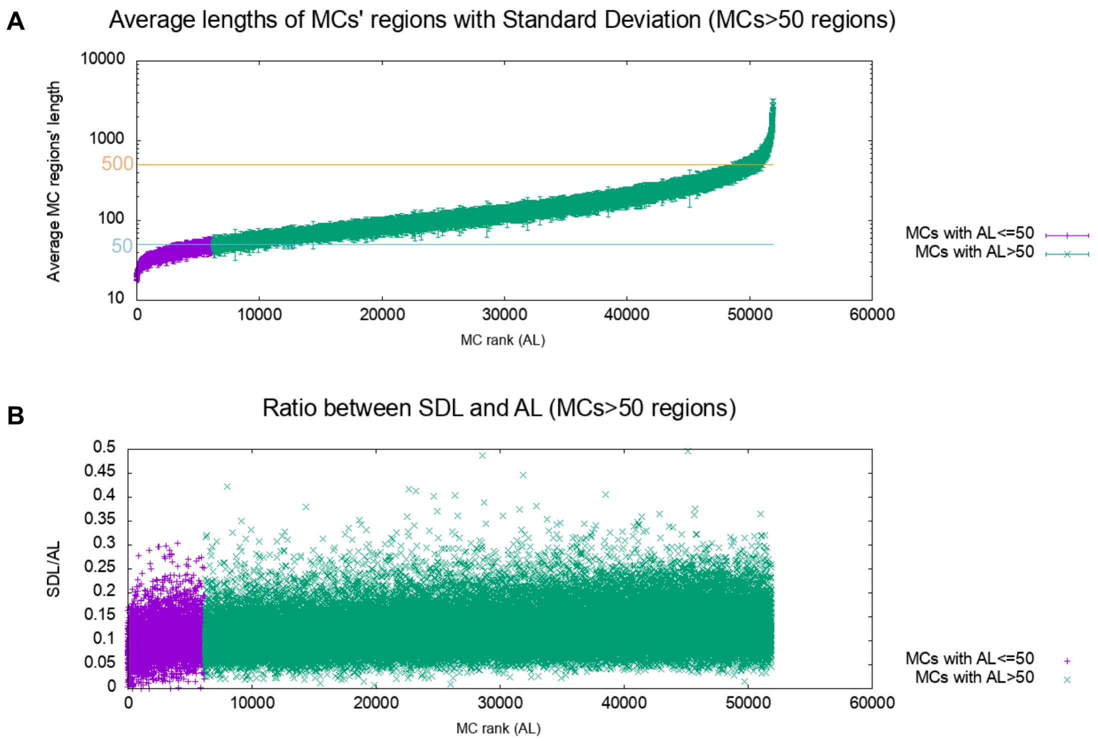


Figure 3.23: A) Average lengths of MCs' regions (AL), with respective standard deviation (SDL) as error-bar. MCs' are sorted with respect their sequences average length, obtaining this rank-size plot. Violet points show MCs with AL<= 50, while green those with AL> 50. B) Metacluster's SDL/AL ratio, with MCs ranked with respect to their AL, as in panel A. Colors as in panel A.

Total MCs generated	210,802
MCs with > 50 regions	51,874
MCs with > 50 regions and AL> 50	**45,679**
MCs with > 50 regions, AL> 50, with UniprotKB v. 2019_08 proteins	**45,670**

Table 3.7: Number of MCs meeting different requirements. We note that in the next subsections we will analyze MCs satisfying the last requirement.

3.3.2 Assigning a Pfam ground truth to UniRef50 proteins

As we already mentioned, the main reference for known protein families used in this work is Pfam (Section 1.5.2). On one hand, we used Pfam annotation to inspect the content of MCs, on the other, we investigated whether MCs covers, and how, Pfam families. Similarly to what we did in Section 3.2.1, first of all we need to annotate all the UniRef50 proteins. Since Pfam offers also UniprotKB annotation, we decided to use directly this information. We note two things: first of all, UniRef50 contain proteins that are not present in UniprotKB (see Section

1.4.1.1); secondly, Pfam does not annotate every UniprotKB release, but only one per Pfam release. Since UniprotKB databases are released at a very faster pace than Pfam's, the UniprotKB v. 2017_07 (the one associated to the UniRef50 database we are using) has no direct Pfam annotation. Moreover, the Pfam version used in Section 3.2, namely v. 31, annotate a UniprotKB version older than the one we use (v. 2016_10 instead of v. 2017_07). Therefore, we moved to the last Pfam version available, specifically Pfam v. 33.0[1], to annotate UniRef50 regions using the respective UniprotKB annotation.

The annotated set of proteins we have, therefore, is the intersection between UniprotKB v. 2019_08 (with Pfam v. 33.0 annotation) and UniRef50 v. 2017_07. While this subset may be partially incomplete, it should provide a useful description of UniRef50 v. 2017_07 Pfam annotation.

UniprotKB v. 2019_08 contains 172,062,311 proteins, of which 132,522,154 are annotated by Pfam v. 33.0, with a sequence coverage of 77.03%. In UniRef50 v. 2017_07 there are 23,531,980 proteins; of these, 18,891,393 (80%) are also part of the UniProtKB v. 2019_08. Within this intersection, 9,315,206 sequences are annotated, with a sequence coverage of 43.9%. When inspecting MCs' content we will take in consideration only these 9,315,206 sequences part of the intersection between the two protein databases.

We removed from the downstream analysis all those Pfam families for which we found less than 100 regions in UniRef50, obtaining 10,631 Pfam families to compare with. Indeed, it is difficult for our algorithm to find such small families. Some further consideration will be done in Section 3.3.4.

3.3.3 Evolutionary consistency of DPCfam MCs

The analysis of MCs' evolutionary consistency uses the measures defined in Section 3.1. The same measures have been used in Section 3.2. In this analysis we will focus more on the global behaviour of such measures.

We recall that, as defined in 3.1.1, the Ground Truth Architecture (GTA) of a region $\mathcal{S}_i$ is the set of Pfam families that, being annotated on that region's protein, are covered at any level by the region (see Figure 3.1). Regions in the same MC may display different GTAs: the most frequent one defines the MC's the Dominant Architecture (DA). DAs can be defined either at a family level or at a clan level.

[1] Pfam v. 33.0 has not been officially released, but it is available at the Pfam ftp website. The official release is actually 33.1, which is the 33.0 version with extra information on SARS-CoV-2 models. Being these beyond of our scope, we opted for version 33.0 as a ”stable” version.

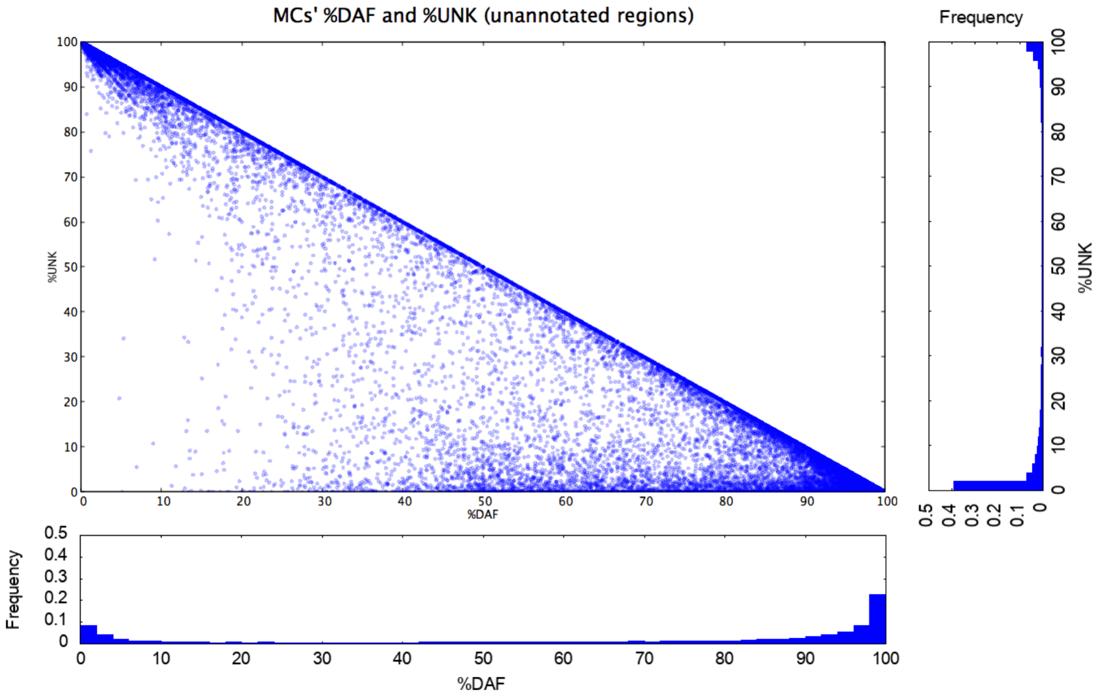


Figure 3.24: %DA and %UNK of the 31,935 MCs with at least one region with Pfam annotation.

We start computing the DA for each MC, at a family and at a clan level. We find that 13,734 MCs (30% of the total) contain only protein regions with no Pfam annotation. These MCs are very interesting to be inspected, being possibly novel families. However, a in-depth analysis of these MCs is beyond the scopes of this thesis, so we will just signal their existence and continue analysing the remaining 31,836 MCs containing at least one Pfam annotated protein region. For such MCs, we can compute the quantities described in Section 3.1.2 nameley: %DAF (percentage of regions having the DA as GTA, at a family level) , %DAC (percentage of regions having the DA as GTA, at a clan level), %DACF (percentage of regions having the DA or a subset of the DA as GTA, clan level) and %DACFA (percentage of regions having the DA, a subset of the DA or a superset of the DA as GTA, clan level). In addition, we can compute for each MC the percentage of regions lacking any Pfam annotation, %UNK (as ”unknown”).

In Figure 3.24 we show a scatter plot of the MCs' %DAF and %UNK, with the respective distributions near to the respective axes. We recall that MCs lacking any Pfam annotation (%UNK=100) are excluded from the analysis. Points are organized in two main clusters, one at the top left and one at the bottom right of the plot. While the second represents MCs with good annotation, the first one

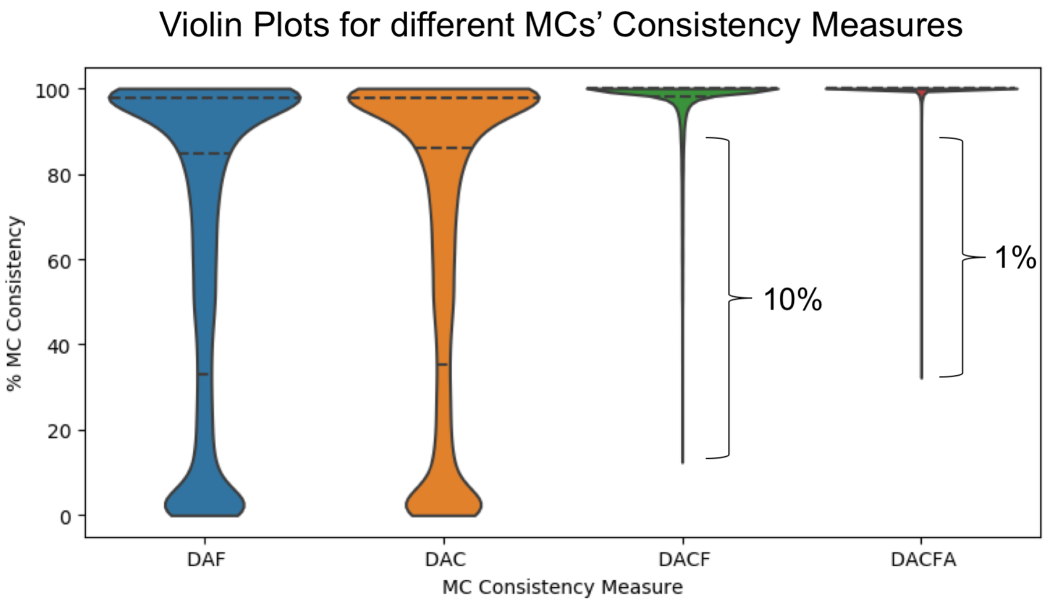


Figure 3.25: Violin plots of the distribution of %DAF, %DAC, %DACF and %DACFA for MCs, where those with %UNK$=100$ are excluded from the analysis. See Section 3.1.2 for definitions. Violin plots are such that their width is proportional to the fraction of MCs with a given value of the respective consistency measure. We note that about 10% of MCs have %DACF< 90, and only 1% have %DACFA< 90.

includes those MCs that are mostly not annotated, being possibly novel families (the few Pfam annotations recorded are most probably noise). There is a third group of MCs, namely those with low %UNK but not optimal %DAF, located around the x-axis: these MCs contain different Pfam annotations. To investigate these last cases, it is more useful to compare the distributions of %DAF, %DAC, %DACF and %DACFA, represented as violin plots in Figure 3.25. We can see that distributions of %DAF and %DAC are extremely similar, and bimodal. %DACF distribution changes notably, and instead of a bimodal distribution we find a single peak around 100: only about 10% of MCs have %DACF< 90. Indeed, when we compute %DACF we include in the computation those sequences lacking part of the DA in their Pfam annotation, besides Pfam un-annotated sequences. The change in the distribution is due to all those MCs that display significant increases from %DAC and %DACFA. Such MCs are, as said in Section 3.1.2 and seen in Section 3.2.2.1, either i)MCs with the potential to increase coverage of existing Pfam families or clans, or ii) MCs constituted of many sequences missing any Pfam annotation, which are good candidates for protein family discovery. This last case, in particular, requires for notably low %DAC and high %DACF: we can estimate the number of these MCs by counting how many display %DAC$<$ 20 and %DACF$>$ 90, which are 20% of the analysed MCs. We recall that also all un-annotated MCs (which have been excluded from this analysis, being such) are candidates for protein family discovery. Case i) requires a large increase in the MC's %DAC to %DACF: for example, we could say that MCs with potential to increase coverage of a Pfam family/clan are those with an increase of at least 30% between %DAC and %DACF, provided that they have a %DACF$>$ 90. We find that 33% of MCs display this feature. Finally, only 1% of MCs (313) have %DACFA$<$ 90, 2% have %DACFA$<$ 95 and 9% have %DACFA$<$ 99. In the analysis of PUA_UR50 and P53_UR50 datasets (see Section 3.2.2.1 and Figure 3.6) some very interesting MCs lied in this regions. While MCs with "low" %DACFA should be classified as evolutionary not consistent according to Pfam classification, in some special cases they could reveal some inconsistency in the Pfam classification itself. The fact that there are very few MCs with this features indicates a reasonable agreement with Pfam classification; however, we believe that those MCs with %DACFA $<$ 95 should be inspected at a deeper level in the future.

We note that, comparing figure 3.25 with Figure 3.6, the shapes of violin distributions displayed are similar, suggesting that the quality of results obtained in this all-to-all analysis are similar to the proof-of-concept experiment.

3.3.3.1 Comparison between MCs and Pfam families boundaries

As done in section 3.2.2.2, to investigate by how much the boundaries of MCs' regions and the respective DAs' regions differ, we use the quantities F^{ext}_{MC} and F^{red}_{MC}, defined in Section 3.1.3 (see Equations 3.2 and 3.1). Here we will add another measure, less specific than F^{ext}_{MC} and F^{red}_{MC}, but still very useful, namely the overlap between a clustered region and its Ground Truth Architecture (GTA). Given a protein region $\mathcal{S}_i$, where $\mathcal{P}_i$ is the protein region covered by its GTA, we can compute the overlap between $\mathcal{S}_i$ and $\mathcal{P}_i$ as:

$$O_{\mathcal{S}_i,\mathcal{P}_i} = \frac{|\mathcal{S}_i \cap \mathcal{P}_i|}{|\mathcal{S}_i \cup \mathcal{P}_i|} \tag{3.3}$$

This definition of overlap is strictly related to the distances used in DPCfam, in particular to $d^{\mathcal{Q}}_{i,j}$ (see Equation 2.6). We can say that $d_{\mathcal{S}_i,\mathcal{P}_i}$ is the distance between $\mathcal{S}_i$ and $\mathcal{P}_i$ (being both on the same protein sequence); therefore

$$O_{\mathcal{S}_i,\mathcal{P}_i} = 1 - d_{\mathcal{S}_i,\mathcal{P}_i} \tag{3.4}$$

Annotated MCs:	**31,935**	
MCs with low %DAC and high %DACF (%DAC<20 and %DACF>90)	20%	Candidates for protein family discovery
MCs with a large increase %DAC to %DACF: (+30% from %DAC to %DACF,%DACF>90)	33%	MCs with the potential to increase coverage of existing Pfam families or clans
MCs with low %DACFA (%DACFA<95)	2%	MCs with evolutionary inconsistencies
Un-annotated MCs (%UNK=100)	**13,734**	Candidates for protein family discovery

Table 3.8: Resume table of MCs, clarifying some characteristics based on values and differences of %DAC , %DACF and %DACFA. Also un-annotaed MCs are reported (for which those quantities cannot be computed and are not considered in the analysis), being in general candidates for protein family discovery.

As we did for F_{MC}^{ext} and F_{MC}^{red}, we can define O_{MC} as the average overlap between MCs regions and its DA (computed on the subset of DA member regions).

In Figure 3.26 we can see, once again, that MCs with a well-defined DA (namely, those for which the %DAF is larger than %UNK, corresponding to the bottom-right section of Figure 3.24) contain mostly high %DAF (points accumulate around %DAF=90-100). Comparing %DAF with the average overlap with the DA, O_{MC}, we can distinguish two regions: A) (top right) contains MCs matching Pfam families (or architectures) both at a %DAF level and at boundaries level; B) (bottom right) contains MCs with good evolutionary consistency, but not matching Pfam families' boundaries. We can expect region A to contain mostly "equivalent" MCs, while "reduced", "extended" and "shifted" will be located in region B (see definitions in Section 3.1.3).

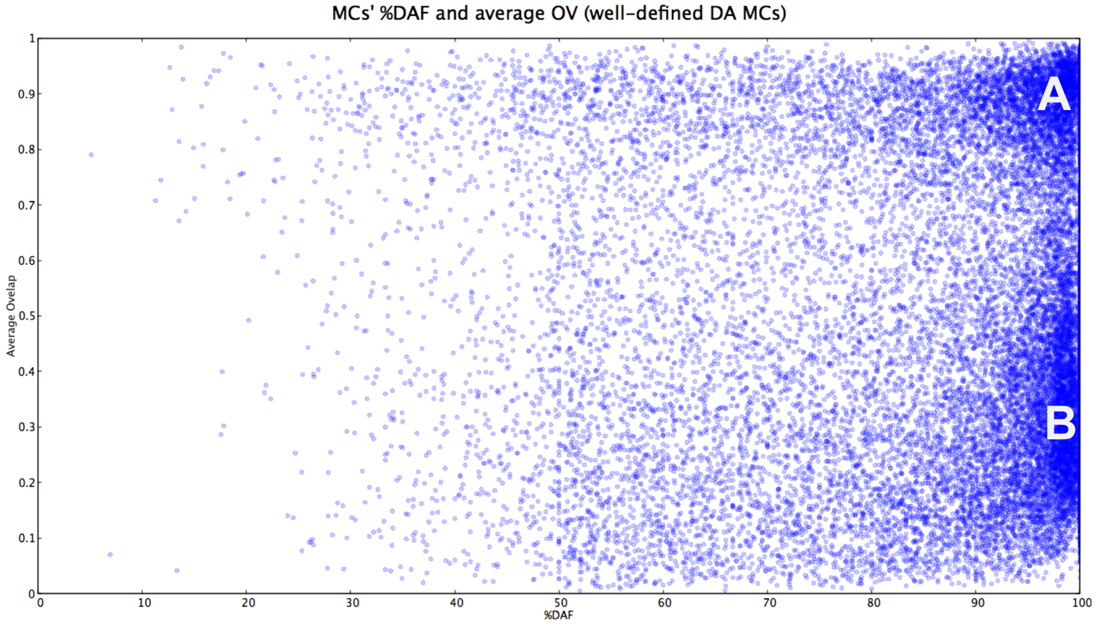


Figure 3.26: %DAF and average overlap (O_{MC}) for MCs with a well-defined DA. A) MCs with good boundaries and consistency with Pfam families/architectures; B) MCs with poor boundaries but good consistency with Pfam families/architectures.

Figure 3.27 shows an histogram of the average overlap O_{MC} computed on the DA at a family and at a clan level, limiting to those MCs with %UNK<%DAF (23,234 MCs) and %UNK<%DAC (23,317 MCs) respectively. We see that there is not a notable difference in defining the DA at a family or at a clan level, as already Figure 3.25 suggested. The two peaks of this bimodal distribution represent regions A ($O_{MC} > 0.7$) and B ($O_{MC} < 0.7$) labeled in Figure 3.26

We then proceed to label each of these MCs as "equivalent", "reduced", "ex-

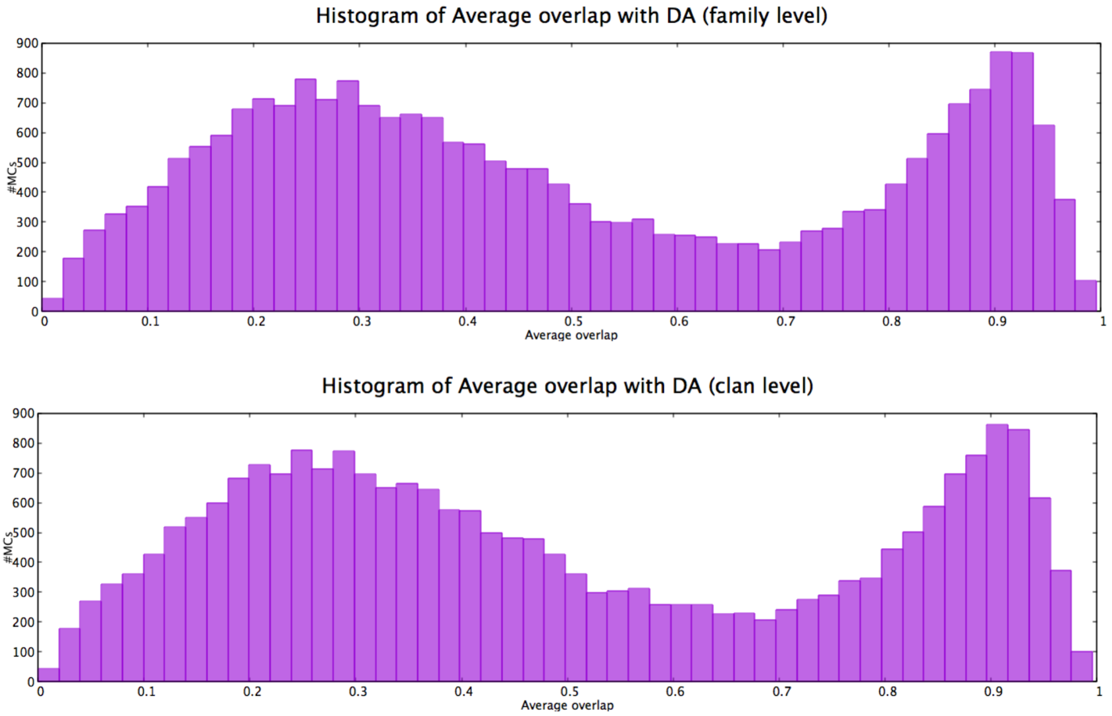


Figure 3.27: Average overlap (O_{MC}) histogram for MCs with a well-defined DA, at a family level (top panel) and at a clan level (bottom panel).

tended” or ”shifted”, as explained in 3.1.3. As we did in the proof-of-concept analysis, we compute F^{MC}_{ext} and F^{MC}_{red} averaging over DA members, where the DA is intended at a clan level. We again exclude those MCs with %UNK>%DAC . Figure 3.28 shows the average ovelap histogram of Figure 3.27 (bottom panel), where columns are colored differently if the MCs contributing are ”equivalent” (yellow, 25% of MCs), ”reduced” (blue, 35% of MCs), ”extended” (pink, 17% of MCs) or ”shifted” (green, 21% of MCs). We note that, as we speculated before, region A of figure 3.26 contains mostly ”equivalent” MCs, while region B is dominated by ”reduced” MCs, namely those capturing a sub-region of known Pfam families. Moreover, equivalent MCs show redundancies on Pfam families only in 2.5% of the cases, namely only 154 MCs are associated to a redundant DA (at a family level), while 6,003 MCs are associated to a unique DA (at a family level).

3.3.4 Coverage of Pfam families by DPCfam MCs

Another question to be addressed to evaluat the performances of DPCfam is the coverage of Pfam families. Here, we investigate if and how we cover any Pfam family present in the database. As explained in Section 3.3.2, in the intersection between

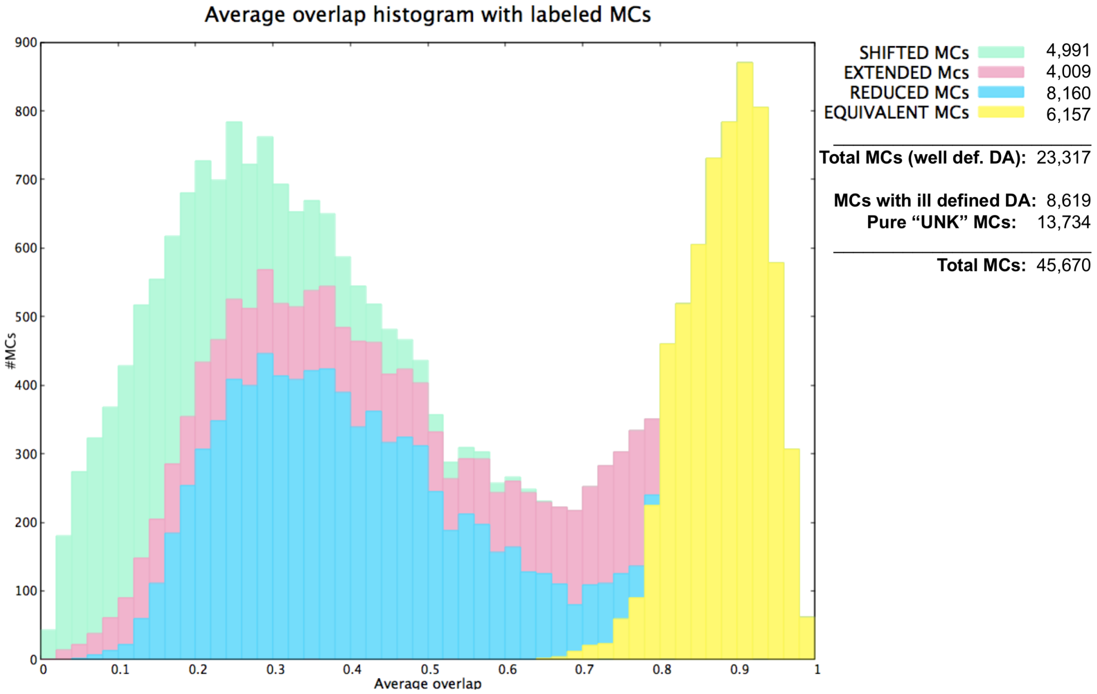


Figure 3.28: Average overlap (O_{MC}) histogram for MCs with a well-defined DA, at a clan level. Columns are colored according to the contribution of each MC category (equivalent, reduced, extended and shifted, see Section 3.1.3 for definitions). Key includes counts of MCs of each category, including a wider count including ill defined (%UNK>%DAC) and pure "UNK" (unannotated, %UNK=100) metaclusters.

UniRef50 v. 2017_07 and the Pfam v. 33.0 annotated proteins in UniprotKB v. 2019_08 there are 10,631 Pfam families represented by at least 100 regions. This is the subset of Pfam families analyzed in this section.

3.3.4.1 Measures to assess Pfam families coverage

Here we will introduce the measures used to assess Pfam families coverage. These are similar to the measures used to inspect MCs' content described in Sections 3.1 and 3.3.3.1, but their definition includes some conceptual differences that need to be explained. In this analysis we focus on two quantities: i) the percentage of the family regions covered by the regions of a MC; ii) the overlap of family regions with MCs' regions. The first quantity measures the percentage of a Pfam family which is also contained in a DPCfam MC, the second one assesses the quality of the boundaries. As done before, together with the overlap we will compute quantities such as F_i^{red} and F_i^{ext} (see Equations 3.1 and 3.2) to classify the nature of the overlap found.

Let us consider a Pfam family, PF, containing n protein regions $\mathcal{R}_i$,

$$\text{PF} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_n\};$$

we compare PF with one of our MCs, containing m regions $\mathcal{S}_j$:

$$\text{MC}_i = \{\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_m\}.$$

First, we estimate the average overlap between PF and any MC i, O_{PF,MC_i}. To do so, we use a definition similar to the average overlap O_{MC} used previously: we search for all couples of regions $\mathcal{R}_i \in$ PF and $\mathcal{S}_j \in \text{MC}_i$ lying on the same protein, and as in Equation 3.3 we compute

$$O_{\mathcal{R}_i,\mathcal{S}_j} = \frac{|\mathcal{R}_i \cap \mathcal{S}_j|}{|\mathcal{R}_i \cup \mathcal{S}_i|} \tag{3.5}$$

O_{PF,MC_i} is the average of $O_{\mathcal{R}_i,\mathcal{S}_j}$, restricted to the the subset of $\mathcal{R}_i$ with a companion regions $\mathcal{S}_j$ in the same protein. Measuring O_{PF,MC_i} alone is not meaningful, since we could obtain $O_{PF,MC_i} = 1$ with a single region in PF perfectly matching a region of MC_i, when the other regions in the MC have nothing in common at all. To solve this, we need to consider the other quantity, that is the number of common sequences between PF and MC_i: we define the $\%\text{MC}_i$ of PF as the percentage of PF's proteins also found in MC_i, regardless with the overlap.

Using these two quantities we define the Dominant MC (DMC) of a Pfam family PF as the MC_i that maximises O_{PF,MC_i}, provided that $\%\text{MC}_i > 50$. From this we define %DMC as the $\%\text{MC}_i$ of the Dominant MC. Since we set a threshold on the $\%\text{MC}_i$s, also $\%\text{DMC} > 50$.

If a Pfam family has a defined DMC, we also compute the F^{PF}_{red} and F^{PF}_{ext} quantities, by applying Equations 3.1 and 3.2 to Pfam regions and the corresponding DMC regions, then averaging over all Pfam regions with a companion DMC region. Then, F^{PF}_{red} tells us how much (in terms of amino-acids) of the Pfam regions are not covered by the respective DMC regions, and F^{PF}_{ext} how much of the DMC regions are not covered by the respective Pfam regions. Again, we can use the four categories introduced in 3.1.3 to find families with an equivalent DMC, families with a reduced (smaller) DMC, with an extended (larger) DMC and with a shifted DMC (see also Figure 3.3, noting that in this case $\mathcal{S}_i$ is the DMC region and $\mathcal{P}_i$ is the PF region).

3.3.4.2 Pfam families coverage using MCs' regions

Here we use the MCs' content to provide an estimate of Pfam families coverage by DPCfam. We stress that MCs' regions are collected in a conservative way (see the filtering procedure in Section 2.2.2.4), and therefore we do not expect to find high coverage rates. However, we believe that this measure is qualitatively meaningful.

We analyse how many Pfam families are covered by a MC. Non-covered Pfam families are those for which there is no MC sharing any common region with it: 402 families with more than 100 elements are not covered, corresponding to 3.7% of such families (10,631). If we consider all Pfam families, including those with less than 100 elements, we find that 4,072 are not covered by any MC, corresponding to the 22.4% of the total (18,189 families). Indeed, as we expected, the coverage of small (<100 elements) Pfam families is notably worse than for larger families. In Figure 3.29 we studied how the coverage of Pfam families varied with respect to their size (blue line). To do so, we computed for each family the best %MC$_i$ as the family's "percentage coverage", with no considerations on the average overlap but to be non-zero. Note that this measure is not equivalent to %DMC. Ranking Pfam families from the largest to the smallest, we compute a rolling mean of their coverage. We find that the average percentage coverage is quite stable for large families (around 60%), and we see that it become worse for sizes smaller than 100-50 elements. We can also see how the average number of non-covered families (orange line in Figure 3.29) rises notably as we reach small sizes.

We then proceed to define for each family the respective Dominant Metacluster (DMC). We note that Pfam families for which it is possible to define a DMC are less than those that have been considered covered in the previous paragraph, since we now ask for a %DMC>50. If without this bound we covered 96.2% of Pfam families (with at least 100 elements), with it we cover 69.1% of them. Still, since we are comparing MCs regions rather than results of profile-HMMs searches, this is a notable number. In Figure 3.30 we show the relationship between the %DMC and the average overlap of Pfam families. We recall that the DMC has been selected to maximise the average overlap, while preserving %DMC>50. We note an accumulation of data points in the top right region of the plot, corresponding to families with a large %DMC and also a large average overlap.

We now apply to Pfam families the same classification we used for MCs in Section 3.3.3.1, by using the DMC as a reference. Therefore, we can distinguish between "equivalent", "reduced", "extended" and "shifted" Pfam families (see Section 3.1.3 for definitions). Points in the top region of figure 3.30 are then "equivalent" families,

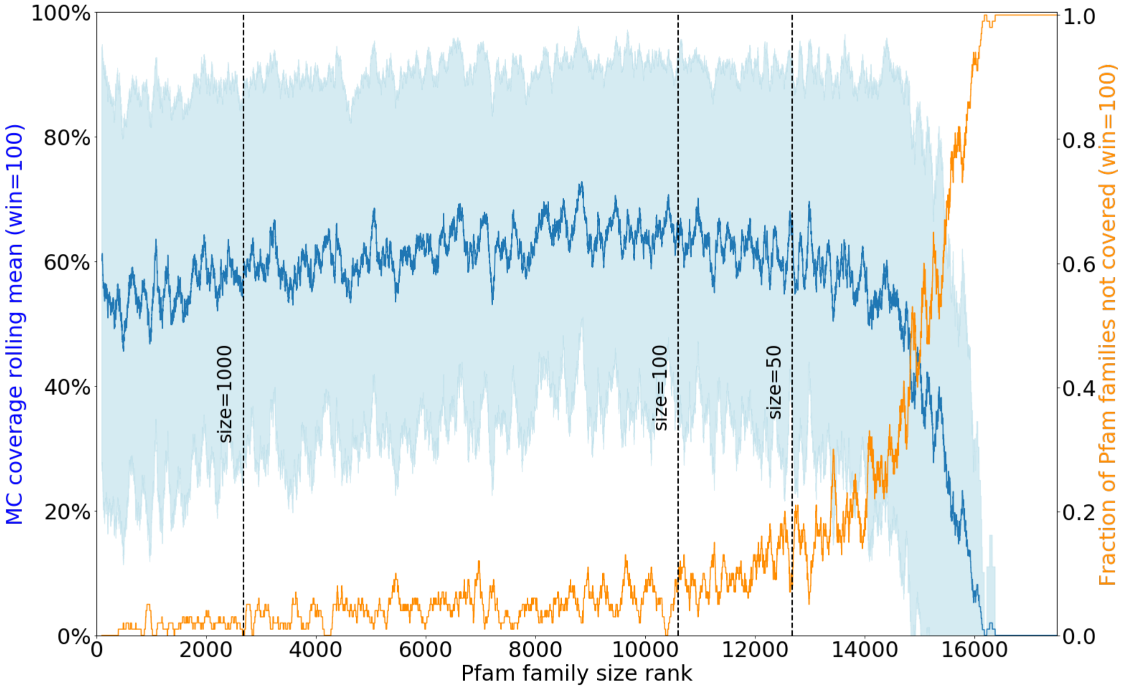


Figure 3.29: Coverage of Pfam families by MCs regions, with respect to Pfam families sizes. Pfam families are ranked with respect to their size (number of sequences in UniRef50), from the largest (148,920 elements) to the smallest (1 element). Coverage is defined as the best %MC$_i$ found for each family (left vertical tics), and it is averaged on a running window of 100 familes (blue line, with error bars as light-blue band). The orange line represents the fraction of non-covered families (right vertical tics), namely those for which it is not possible to find any metacluster covering the family; also in this case we show the average over a window of 100 elements).

namely families for which it is possible to find a MC with a good coverage which also reproduces well its boundaries. In Figure 3.31 we present an histogram of the average overlap of Pfam families with their DMC. Columns are coloured with respect to families classification. We see that the overlap distribution has a maximum at high values of $O_{PF,DMC}$, where "equivalent" families are found. Of all Pfam families with a DMC (7,350), almost half have an "equivalent" DMC (46.9%). Only 757 families (10.3%) are represented by shorter DMCs (reduced), and even less by a "shifted" DMCs (436, 5.9%). On the other hand, a large number of families have a wider ("extended") DMC (2,710, 36.9%).

3.3.5 HMMs models of the metaclusters

DPCfam's MCs are built in a conservative way: the method to generate them aims to collect together homologous protein regions, but not "all" of them. Indeed,

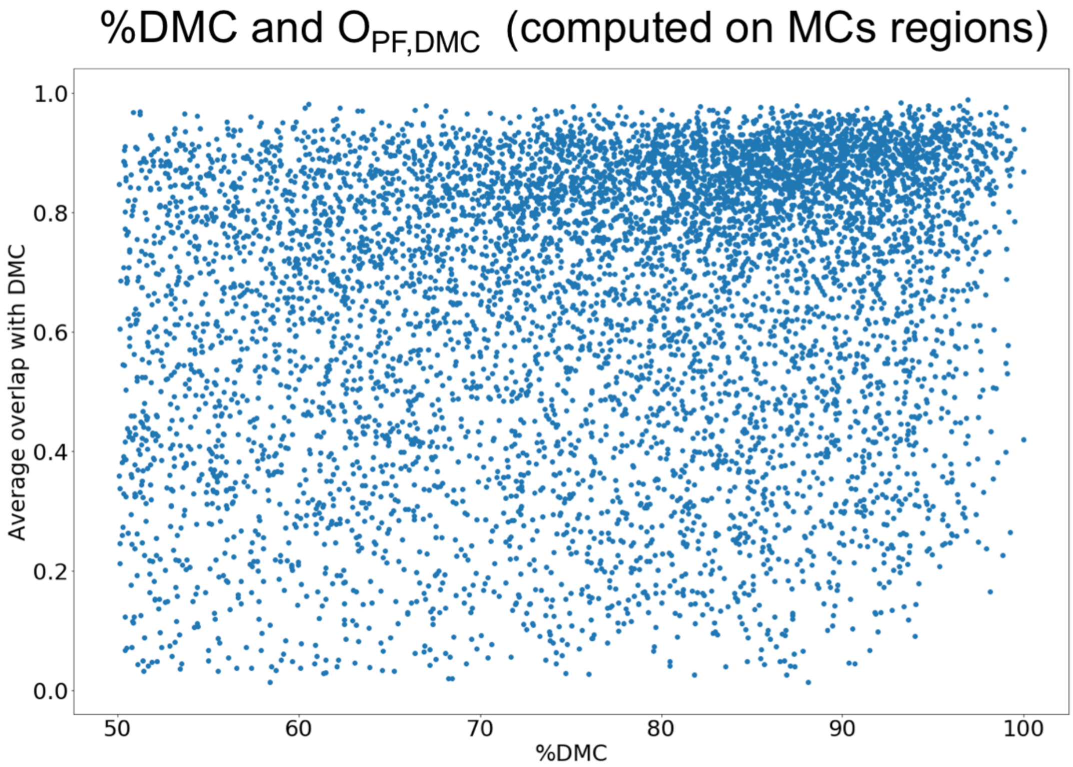


Figure 3.30: Scatter plot of Pfam families' %DMC as a function of $O_{PF,DMC}$. Results obtained using MCs regions to define DMC and compute $O_{PF,DMC}$.

during the clustering process we discard from a MC a large quantity of protein regions that could be still homologous to its elements, but for our criteria are not enough reliable. Thus, DPCfam's MCs cannot be considered as the final "families" found, but rather they should be considered as families' seeds. On par with what protein families databases do (see Section 1.5), we use MC's regions to generate profile-HMMs to extend our "families" definition (see Section 1.4).

For each MC, we build its respective profile-HMM with the following procedure: first we reduce MCs' redundancy to 60 P.I. (using Cd-Hit [37]), then we automatically build MSAs by using MUSCLE [31]. Before building MSAs, we allow for a maximum of 5,000 sequences in each of them: when we have too many, we select randomly 5,000 representative sequences. From these MSAs we build the respective profile-HMMs using hmmbuild, v. 3.1b2. Finally we use profile-HMMs to search for hits in UniRef50: a "hit" corresponds to a protein region that is found to match the profile-HMM. We use hmmsearch (v. 3.1b2) imposing an E-value domain threshold of 0.01 and an E-value protein threshold of 1 (following the standard values used in

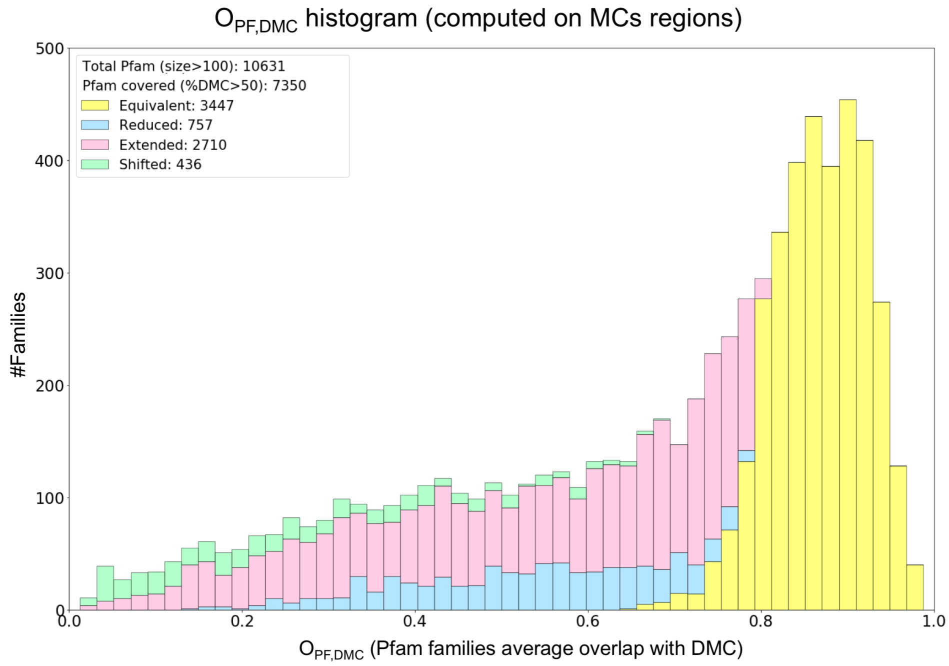


Figure 3.31: Histogram of Pfam families' average overlap with DMC ($O_{PF,DMC}$), with columns colored according to the family classification (equivalent, reduced, extended and shifted, see Section 3.1.3). Results obtained defining the DMC and computing $O_{PF,DMC}$ using MCs' protein region.

the hmmsearch website).

3.3.5.1 Evaluating DPCfams' profile-HMMs quality

Our first question is whether or not DPCfam's profile-HMMs are reliable. Such a question is not easy to answer: for example, one could think that profile-HMMs producing hits with very good E-values are the best ones, but this is a naive interpretation. Being profile-HMMs used to generalise a seed MSA over a protein database, one would like to find not only hits evolutionarily close to the model (reaching very low E-Values, namely very good ones), but also those that are less close to the original seed, while still meaningful (with higher E-values). To understand how difficult it is to define a generally "meaningful" hit obtained with profile-HMMs, we note that in Pfam each model has a specific gathering threshold [58], namely a threshold on the bitscore (derived of the score) set to avoid false positives, besides overlap with other families, when using hmmsearch. Such threshold is family-specific and manual curation plays a relevant role in defining it. This procedure cannot be easily applied on DPCfam's profile-HMMs, for which, by now, we use a standard E-Value threshold. Overall, E-values averages or similar quantities are not a sufficient measure of quality for a profile-HMM.

First of all, we study the general statistical properties of profile-HMMs, and we compare them to those of Pfam's models. Such a study does not require to run hmmsearch and is therefore free from E-value/gathering thresholds. The hmmer suite offers a specific program to compute profile-HMMs statistics, hmmstats [59]. Some interesting quantities computed by hmmstat are (definitions are directly taken form the manual):

- **eff nseq** - The effective number of sequences that the profile was estimated from, after HMMER applied an effective sequence number calculation such as the default entropy weighting.
- **relent** - Mean relative entropy per match state, in bits. This is the expected (mean) score per consensus position. This is what the default entropy-weighting method for effective sequence number estimation focuses on, so for default HMMER3 models, you expect this value to reflect the default target for entropy-weighting.
- **compKL** - Kullback-Leibler distance [60] between the model's overall average residue composition and the default background frequency distribution. The higher this number, the more biased the residue composition of the profile is.

We start analysing "relent": as specified in the description, this quantity is the one optimized by hmmbuild: in particular, the target for entropy weighting is set to relent= 0.59. Together with this parameter, also the minimum relative entropy contibutes in the optimization: its value is bounded to a minimal value (45.0 bits) which "has the effect of making short models have higher relative entropy per position" (from [59], hmmbuild). Therefore, as a minimal sanity check we should expect that i) short profile-HMMs should have relent>0.59, growing as the profiles lengths decrease; and ii) other profile-HMMs should reach the optimal relent value, 0.59. This property is confirmed and can be seen in Figure 3.32 both for DPCfam's models and Pfam-A's models, as a function of the profile-HMMs length (M). We note that the general behaviour described above is replicated, but some of DPCfam's profiles show a smaller relent value than the minimal value expected (about 30 profiles). This may signal a bad quality for the original MSAs, displaying a too high level of noise. Still, the fraction of DPCfam's profile-HMMs for which this happens is very small.

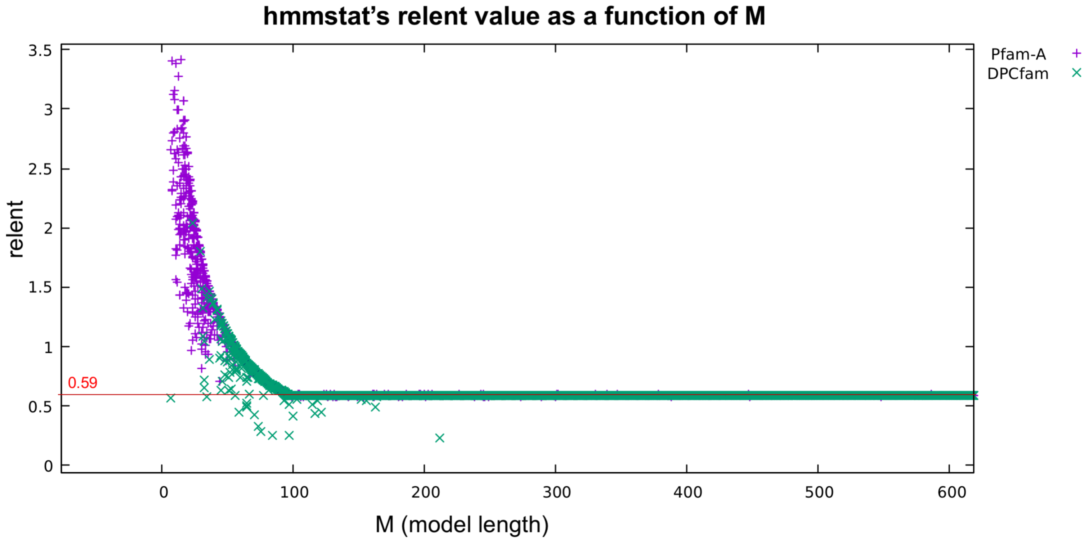


Figure 3.32: Values of relent (mean relative entropy per match state, in bits) for DPCfam and Pfam-A profile-HMMs, in function of the model's length (M). We show cases with $M < 600$

In general, it is better to compare profile-HMMs of similar lengths (M). We divide profile-HMMs in three categories: 50 <M< 100 , 100 <M< 200 and 200 <M< 500, and we compute the respective distributions of eff nseqs, both for Pfam and DPCfam models (Figure 3.33 A). As an interesting result, we note that DPCfam's profile-HMMs tend to have a larger number of effective sequences: this is reasonable to us if we consider that Pfam-A's seeds are defined by human curation, while DPCfam's MCs are obtained clustering UniRef50 sequences, and cleaning the results by redundancy reduction (less than 60 P.I. among sequences of the same MC).

Similarly, we compute the distributions for compKL (Figure 3.33 B). This quantity evaluates how much the model's overall average residue composition is close to the default background frequency distribution. Small values accounts for profiles that are basically reproducing the background frequency, while at higher values models have large compositional biases. From the plots we can see that Pfam-A models and DPCfam models does not show significant differences in the distirbutions of these values, in the respective ranges of M. In conclusion, from HMMs statistical properties we can basically say than DPCfam and Pfam-A models show similar characteristics, suggesting that there is no striking difference between manually curated models and automatically-generated DPCfam models, at least from the quantities estimated from hmmstat.

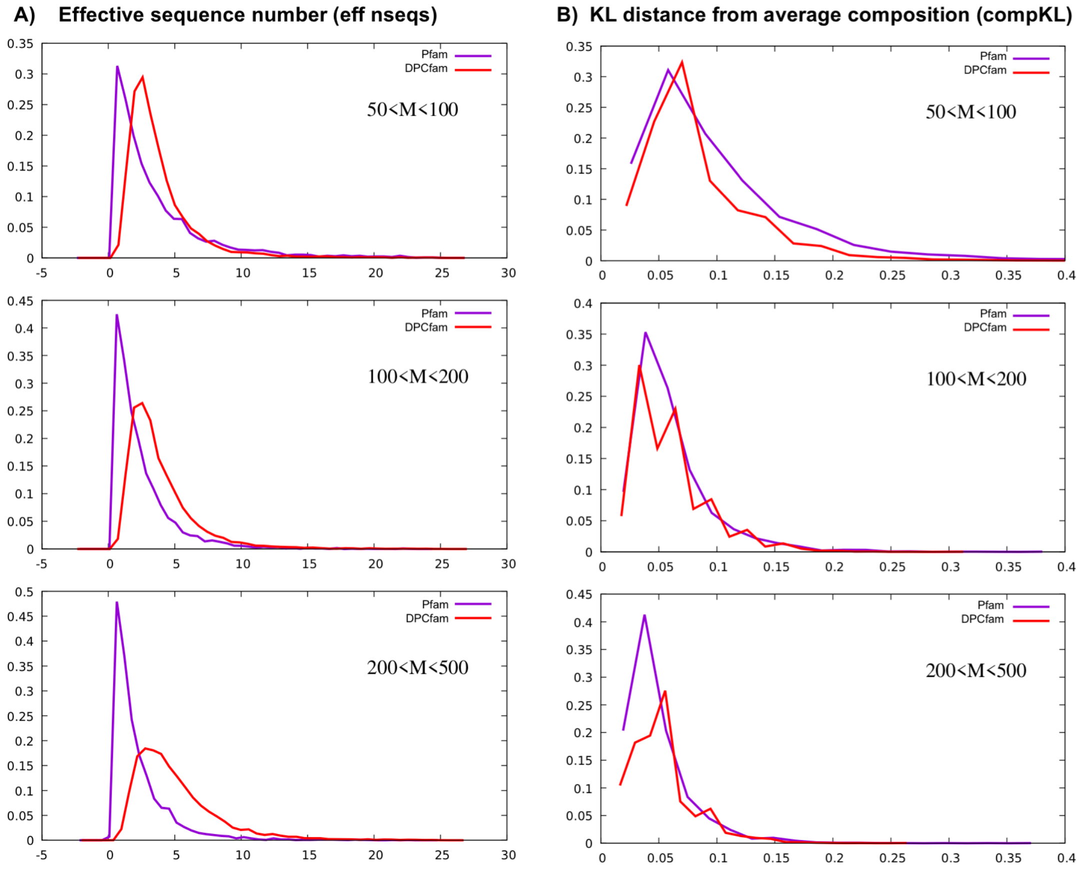


Figure 3.33: Distributions for A) effective sequence number (eff nseqs) and B) KL distance from average composition (compKL), for Pfam-A profile-HMMs and DPCfam profile-HMMs. Models are divided in categories (top, middle and bottom panels) according to their lengths (M).

3.3.5.2 Comparing DPCfam to Pfam-A profile-HMMs hits for ”equivalent” metaclusters

In the particular case of MCs matching to a Pfam family at an ”equivalent” level (see Section 3.1.3), we can directly compare their profile-HMM results with those obtained with the respective Pfam-A model. In this case we search using Pfam-A profile-HMMs using the same threshold as we did for DPCfam's models (thus, ignoring the profile-specific gathering threshold). This allow for a fairer comparison between the two sets of matches.

As a first example, in Figure 3.34 we report results for 19 MCs randomly chosen among ”equivalent” MCs with a single-family architecture. We collect all hits, in terms of proteins found, and we divide them between common hits, hits found only by DPCfam and hits found only by Pfam-A: in panel A we display all results,

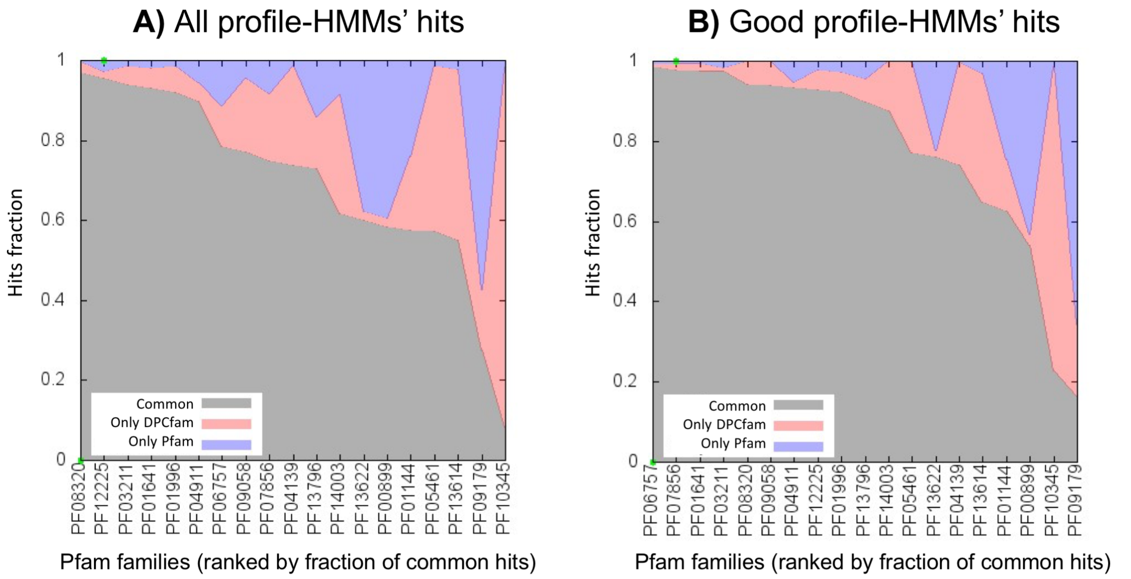


Figure 3.34: Comparison between profile-HMMs hits obtained from Pfam-A models and DPCfam models of a subset of 19 ”equivalent” MCs. Each column corresponds to a MC-Pfam family couple, where gray regions correspond to proteins found by both models, light-red regions correspond to proteins found only by DPCfam's models and blue regions correspond to proteins found only by Pfam's models. The fraction is computed over the total number of proteins found by both models. Pfam families on the x-axis are ranked according to the fraction of common hits. Panel A: all hits; Panel B: ”good” hits (sequence E-value<0.0001)

while in panel B we restrict ourselves to ”good” hits, namely with sequence E-value <0.0001. For almost every family the fraction of common hits is notably high (larger than 0.6 of all hits), and restricting to good hits increases this fraction. In different cases, Pfam or DPCfam display higher number of hits. At the right side of the plots we find two ”problematic” families, corresponding to PF09179 and PF10345. Pfam family PF10345 has a very small seed (19 elements) while the respective MC found 228 elements, with about 20% of unannotated regions. Such a difference in seeds' sizes can easily lead DPCfam's model to cover more protein regions than Pfam's. At this level, we cannot say anything about which of the two models can be considered better than the other. Pfam family PF09179 displays the opposite situation: our MC found only 118 regions, while Pfam's seed counts over 210 regions. It is then probable that DPCfam found a subset of the family, and cannot extend the annotation to regions of the protein space that are not represented in the MC.

Figure 3.35 shows the sequence E-value pseudo-distributions for hits obtained using the Pfam family profile-HMM or DPCfam's MC profile-HMM. Panel E and

F shows the two bad-performing cases described above, while panels A, B C and D show other better cases. We can see that in the first four panels E-value distributions of the two models are very similar.

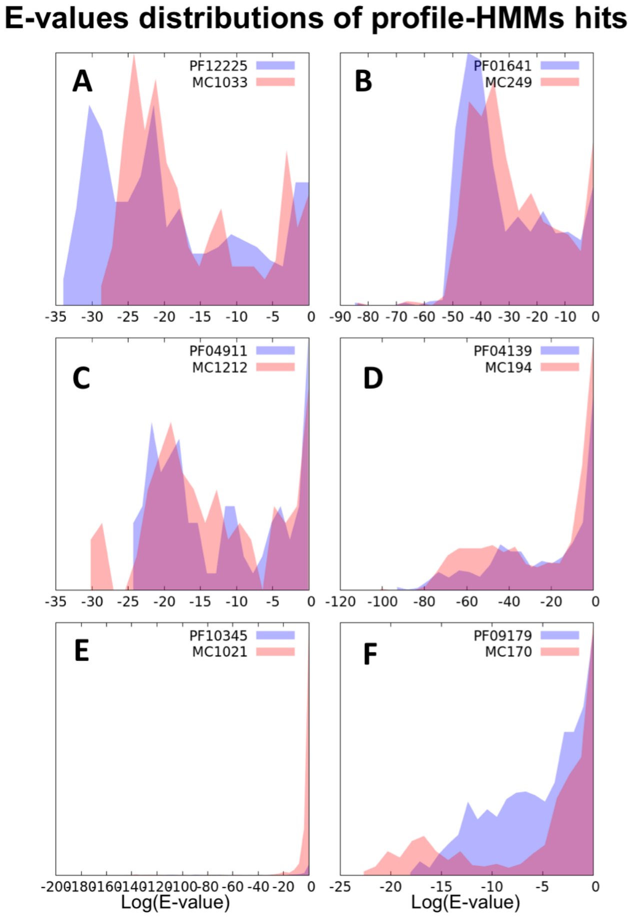


Figure 3.35: Examples of E-value pseudo-distributions of ”equivalent” MC’s profile-HMM hits, together with the same obtained from the respective Pfam family’s profile-HMM. Pseudo-distributions are non-normalized distribution, in order to show the difference in hits between the two models. Hits have been obtained searching with hmmsearch on UniRef50, with standard parameters (sequence E-value$<$ 1, domain E-value$<$ 0.01). Blue areas show Pfam’s models’ E-values, while red areas show the respective MC profile-HMMs’ results.

Besides the 19 example MCs shown above, we analyze all ”equivalent” MCs mapping to a single Pfam family ($\sim 5,000$ MCs). In Figure 3.36 A, the black line shows the fraction of hits obtained by both the MC model and the respective Pfam family model, computed with respect to all hits found by both models. In most of the cases ($\sim 80\%$) the two models have a fraction of common hits larger than 0.6. Restricting to good hits (E-value < 0.0001, Figure 3.36 B), further enlarges the fraction of common hits. The blue line and red line show the fraction of hits obtained only by the MC's derived profile-HMMS and Pfam's profile-HMMs. Values are sorted independently from the largest to the smallest. We note that MC's profile HMM tend, on average, to find more hits than Pfam's models. This is somehow expected, considering that MC's MSA tend in general to include more sequences that Pfam's curated seeds.

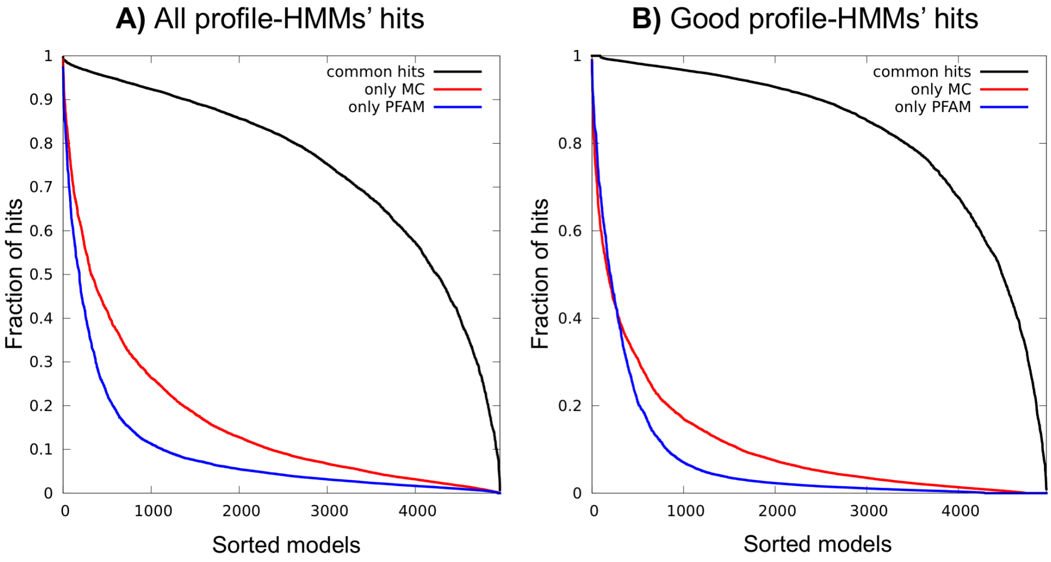


Figure 3.36: Comparison between profile-HMMs hits obtained from Pfam-A models and DPCfam models of "equivalent" MCs, mapping to a single-family Dominant Architecture. The black line shows the fraction of common hits between the two models, sorted; the red line shows the fraction of hits found only by the MC's models, sorted; the blue line shows the fraction of hits found only by the Pfam models. Fractions are computed with respect to the sum of all hits found by both the MC and the Pfam model. X-axis represents MC-Pfam family couples, sorted differently for lines of different colors. Panel A: all hits; Panel B: "good" hits (sequence E-value<0.0001)

In Figure 3.37 we plot, for each couple of MC and Pfam family, the fraction of common hits as a function of the MC's percentage of Dominant Architecture, specifically %DAF. We recall that %DAF is the percentage of regions in the MC whose Ground Truth Architecture (GTA) matches the MC's Dominant Architecture (DA) at a family level. MCs with small %DAF include therefore also sequences that, according to Pfam, are not part of the DA's family. In Panel A we notice that the vast majority of MCs which are equivalent to a Pfam family display a %DAF between 90 and 100. The fraction of common hits of the respective profile-HMMs accumulate around 0.9, but we note that low fractions are possible also for large values of %DAF. However, good hits (Panel B) display a significantly larger average fraction of common hits. MCs with a large %DA and a relatively small fraction of common hits are most probably MCs smaller than the respective Pfam family's seed: in this case only a subset of the family can be represented, which lowers the fraction of common hits while still the MC's DA is large (see the case of PF09179 in the previous example).

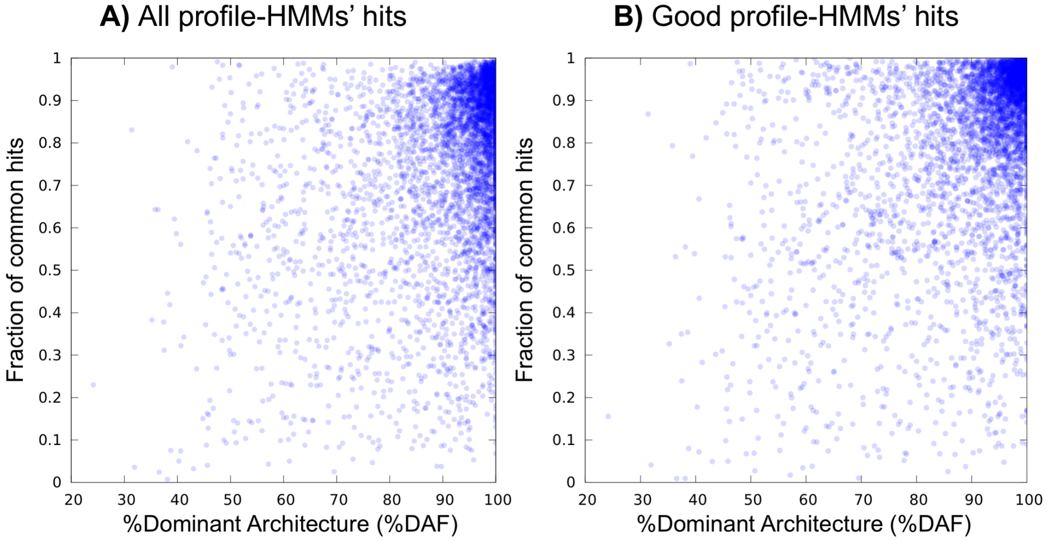


Figure 3.37: Comparison between the fraction of common hits found by both an "equivalent" MC profile-HMM model and the respective Pfam family model (y axis) and the %DAF of the MC (x axis). MCs considered maps to a single family. Panel A: all hits; Panel B: "good" hits (sequence E-value<0.0001)

Analysis done so far cannot be reproduced straightforwardly on non-equivalent MCs. However, we believe that there is no striking reason to say that MCs found to be equivalent to a Pfam family should have a different quality to non-equivalent MCs. To support this idea, we computed for a set of profile-HMMs derived from MCs of different categories the fraction of good hits they found (E-value<0.0001). Figure 3.38 shows the distributions of such fraction, for different model lengths (M). Such distributions are overall very similar within the range of M selected. This makes us believe that the quality level found in "equivalent" profile-HMMs can be found also in other categories of MCs. This is particularly relevant for "unknown" MCs, namely those MCs with no Pfam family associated that may represent novel families.

To sum up, in the particular case of "equivalent" MCs associated to a single Pfam family, the agreement between DPCfam's and Pfam's models is notably high.

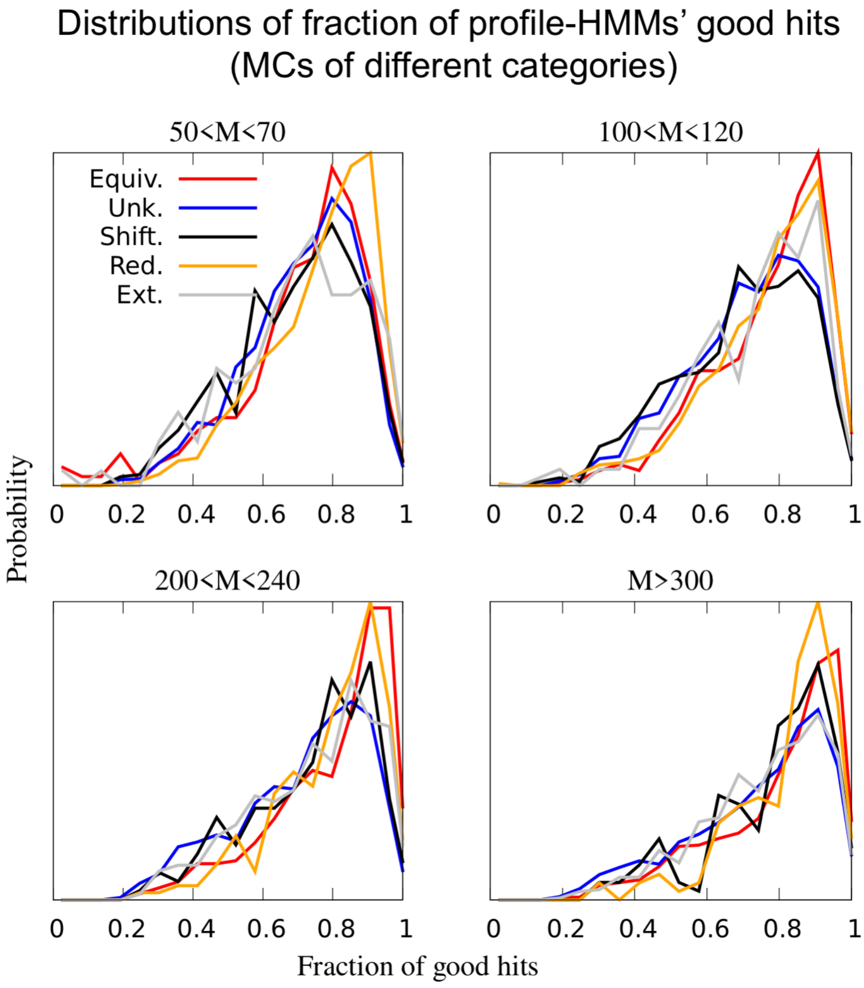


Figure 3.38: Distributions of fraction of profile-HMMs's good hits (E-value<0.0001). Distributions have been divided by models' length (M) and MC's category. Here, red lines represent "equivalent" MCs, yellow lines represent "reduced" MCs , grey lines represent "extended" MCs, black lines represent "shifted" MCs and blue lines represent "unknown" MCs. The latters are MCs with %UNK=100. Category definitions are described in Section 3.1.3 .

The relatively few cases where either DPCfam's models or Pfam's models find more hits than the others are possibly due to differences in the sizes of the respective MSA's used to generate the profile-HMMs. In most cases, the two models obtained with two different methods produces similar results.

3.3.5.3 Protein databases coverage

An important measure to evaluate a set of protein families is the number of proteins that can be ”annotated” by the set, namely which are found as hits by at least one profile-HMM. The percentage of annotated proteins in a database is commonly called ”sequence coverage”[2]. Each new Pfam release reports UniprotKB coverage as one of the useful measures to evaluate the family database’s progression. For example, Pfam-A v. 31.0 covers 75.5% of UniprotKB’s proteins (v. 2016_10), while the more recent version 33.0 covers 77.0% of UniprotKB’s protein (v. 2019_08). To effectively compare Pfam’s coverage to DPCfam’s, we must restrict to the protein database used by both. We note that if we restrict to UniprotKB’s proteins that are also part of the UniRef50 (v. 2017_07) database the coverage of Pfam-A v. 33.0 drops to 43.9%.

Protein Database	**DPCfam HMMs hits**	**Pfam-A UniprotKB annotation**	**Pfam-A HMMs hits**	**Pfam-A HMMs hits (restricted)**
UniProtKB v. 2019_08	//	77.0%	//	//
UniRef50 v. 2017_07	58.9%	//	54.4%	51.2%
UnirotKB ∩ UniRef50	58.6%	43.9%	54.1%	51.0%

Table 3.9: Protein coverage of DPCfam and Pfam-A of different protein databases. DPCfam profile-HMMs hits (second column) and Pfam-A profile-HMMs hits (fourth colum) have been obtained by running hmmsearch on UniRef50 (v. 2017_07) with sequence E-value < 1 and domain E-value < 0.01. In fifth column we restrict the computation to Pfam families which are ”large enough” (at least 50 a.a. of regions’ average length and 100 elements). Third column is the official annotation of UniprotKB (v. 2019_08) released by Pfam. Using Pfam v. 33.0 and hmmsearch v. 3.1b2.

We compare the coverage of profile-HMMs hits of Pfam-A families and DPCfam families, considering three different protein databases: UniprotKB (v. 2019_08), UniRef50 (v. 2017_07) and the intersection between these two. We note that it is not always possible to define the coverage on the first two databases, which makes the intersection between UniprotKB and UniRef50 the actual reference.

DPCfam’s profile-HMMs covers 13,866,000 proteins, corresponding to 58.9% of all UniRef50 proteins. If we restrict to the subset of proteins both in UniRef50 and UniprotKB (see Section 3.3.2) DPCfam profile-HMMs cover 58.6% of proteins, while Pfam-A annotation (see Section 3.3.2) covers 43.9% of proteins. For a better comparison of the coverage, we run hmmsearch with the same parameters used in DPCfam models using Pfam-A models (excluding then family-specific gathering thresholds). With this method, Pfam-A covers 54.4% of UniRef50 proteins, and 54.1% of proteins in the intersection between UniprotKB and UniRef50. We recall that we exclude from the analysis DPCfam's MCs with average length smaller than 50 amino acids; moreover, it is difficult for the protocol to find families with few representatives. Therefore, a fairest comparison should exclude Pfam families which are too short (less than 50 amino acids as average length of their members) and with too few representatives (less than 100 elements in Pfam-A annotation). With these restrictions, Pfam-A's profile-HMMs covers 51.2% of UniRef50 (and 51.0% of the intersection of UniRef50 with UniProtKB). See Table 3.9 for a summary of these results.

3.3.5.4 Pfam Families' coverage using profile-HMMs' hits

As we did for MCs in Section 3.3.4.2, we computed the coverage of Pfam families. Here we used MCs' profile-HMMs' hits in place of MCs' regions. First, we analysed how much DPCfam's models covered Pfam families. In this case, we do not cover 420 Pfam families with more than 100 elements, while we do not cover 5,519 families with any number of elements. We note that these numbers rose a bit from the same numbers computed using MCs regions. This is most probably due to the "trimming" of protein regions done by hmmbuild, which removed some non-relevant amino acids actually covering a bit of Pfam families which where, otherwise, not covered in any other way. Such a coverage was spurious, and has been correctly removed when building the profile-HMM. Comparing Figure 3.39 to Figure 3.29, we see that the coverage of Pfam families is notably increased (from an average of 60% to 80%). Moreover, the size effect for familes with less than 100 elements becomes more relevant. This indicates that, even if from Figure 3.29 it seemed that DPCfam was somehow able to detect such small families, the resulting profile-HMMs models are of a poor quality. We continue, then, to exclude from the further analysis Pfam families with less than 100 elements.

We then compute for each Pfam family the DMC, the %DMC and the average overlap $O_{PF,DMC}$. In Figure 3.40 we put in relation %DMC and $O_{PF,DMC}$. If we compare to the results obtained on MCs' regions (Figure 3.30) we see a notably different distribution, here with much more points located around large values of %DMC. This effect is due to the large increase of regions covered thanks to profile-HMMs. This leads i) to a general increase in %DMC for Pfam families, which have more chances to be better covered, and ii) to a larger number of Pfam families

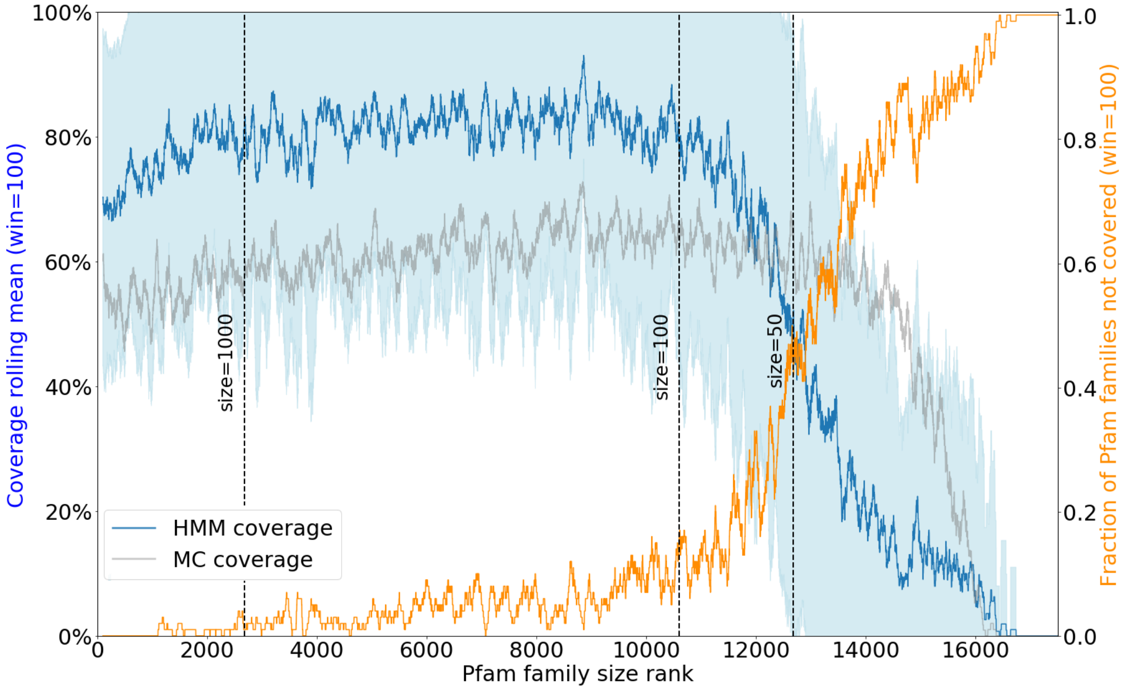


Figure 3.39: Coverage of Pfam families by MC's profile-HMMs regions, with respect to Pfam families sizes. Pfam families are ranked with respect to their size (number of sequences in UniRef50), from the largest (148,920 elements) to the smallests (1 element). Coverage is defined as the best %MC$_i$ (computed using profile-HMMs' hits as MC regions) found for each family (left vertical tics), and it is averaged on a running window of 100 familes (blue line, with error bars as light-blue band). The grey line shows the coverage computed only on MC's regions (Figure 3.29) for a comparison. The orange line represents the fraction of non-covered families (right vertical tics), namely those for which it is not possible to find any metacluster covering the family; also in this case we show the average over a window of 100 elements).

covered by a DMC. Using MCs' regions, 69.1% of all Pfam families (with more than 100 regions) had a DMC, using profile-HMMs hits 84.5% of such Pfam families have a DMC.

The effect of increasing the coverage using profile-HMMs hits can be seen also in Figure 3.41. This histogram, compared to that in Figure 3.31 shows a larger number of "equivalent" families: 49.0% of all 8,988 found with a DMC (it was 46.9% over 7,350 with MCs). Moreover, we find a notably smaller number of "extended" families: 27.6%, namely 2,477 (we had 2,710 extended families in Figure 3.31, 36.9% of the total in that case).

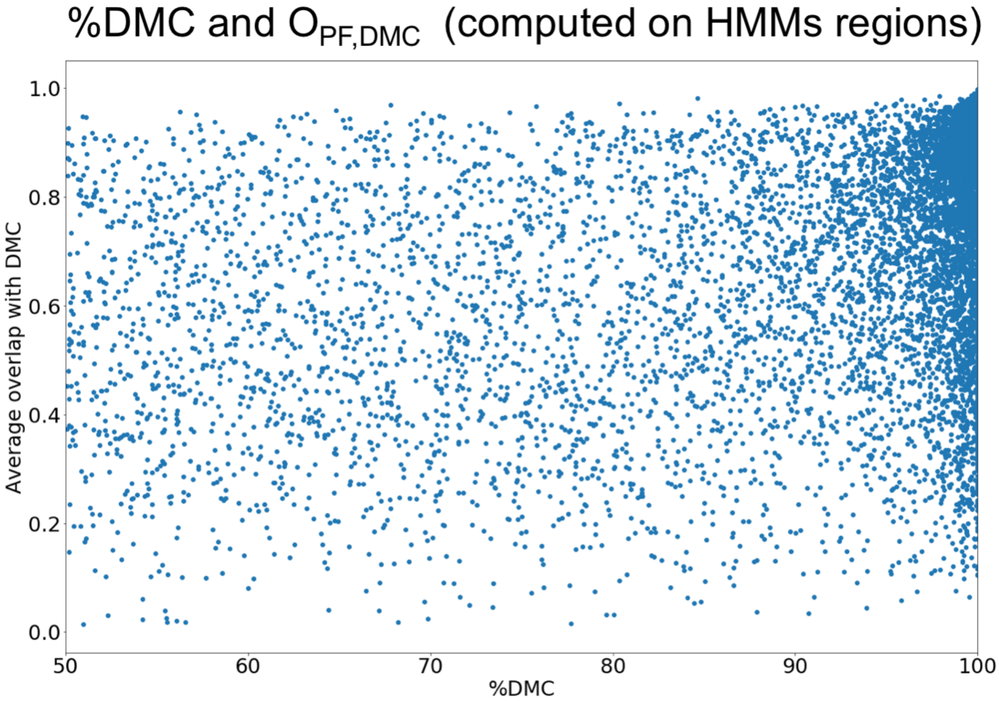


Figure 3.40: Scatter plot of Pfam families' %DMC compared to $O_{PF,DMC}$. Results obtained using MCs' profile-HMMs' hits to define the DMCs and compute $O_{PF,DMC}$.

3.3.6 A discussion of the all-to-all clustering results

The clustering of all UniRef50 database by DPCfam produced about 40,000 metaclusters, of which about a third are classified as "unknown", namely their regions do not overlap with any Pfam family region: these MCs are very interesting candidate to discover new protein families. Among the other MCs, about 23,000 have sequences that contains a reasonable number of protein regions mapping to the same Pfam family (or architecture), 6,000 of which replicate such Pfam annotation with notably good boundaries. Results show a good agreement with the Pfam classification: Pfam families are well covered by DPCfam's MCs (Figure 3.39), and abour 30% of all Pfam families are represented by a DPCfam MC with a good family coverage and high-quality boundaries (see Figure 3.41). DPCfam profile-HMMs covers a larger number of UniRef50 sequences than Pfam-A models (58.9% DPCfam, 51.2% Pfam, see Table 3.9). We note that it is improbable that all UniRef50 proteins actually contain a protein family, so that the best possible sequence coverage is smaller than 100%: therefore, the difference between DPCfam and Pfam protein coverages is more relevant than it appears. Results obtained are encouraging about

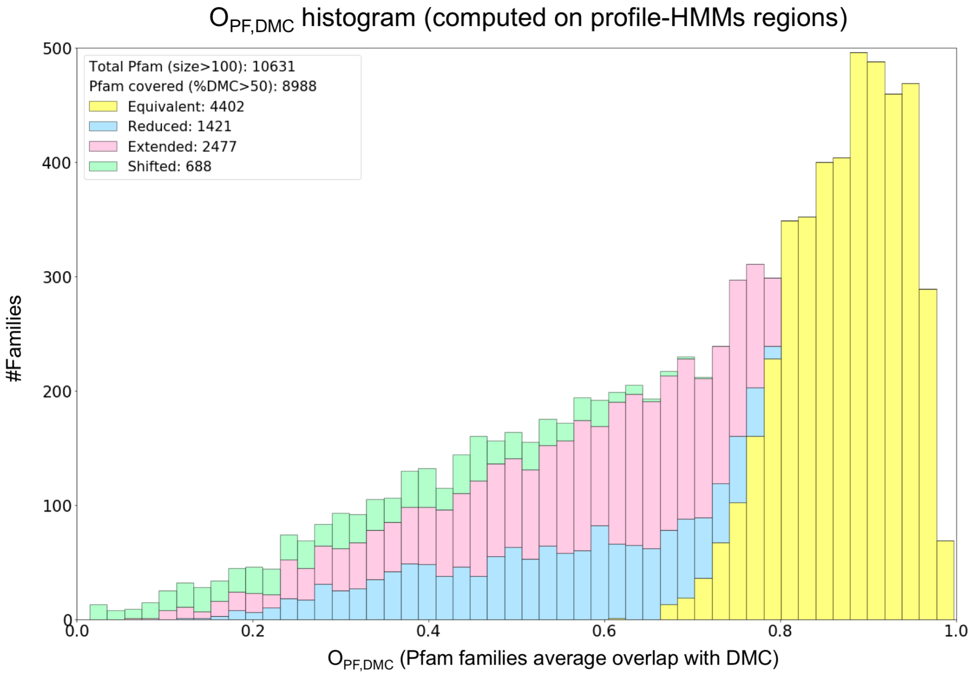


Figure 3.41: Histogram of Pfam families' average overlap with DMC ($O_{PF,DMC}$), with columns colored according to the family classification (equivalent, reduced, extended and shifted, see Section 3.1.3). Results obtained using MCs' profile-HMMs' hits to define the DMCs and to compute $O_{PF,DMC}$

the effectiveness of DPCfam protocol. We do not expect DPCfam automatically generated clusters to completely replicate Pfam's classification, considered the very different methods used to define families in the two approaches. As we learned in the proof-of-concept experiment (Section 3.2), situations where DPCfam and Pfam classifications disagree are not straightforward to evaluate. Thanks to the measures described in section 3.1, we are in principle capable to identify those MCs that could help in extending Pfam family classification or whose apparent disagreement with the latter hides a possible novel family definition (see Table 3.8).

Chapter 4

Conclusions and Perspectives

In the characterization of proteins, protein families play an essential role. Protein regions that are members of the same family are homologous, stemming from a common ancestor. If structured, they typically share a similar fold. Moreover, they may share similar functions. For a protein region, family membership can be determined using only sequence-based information. Novel protein sequences can be therefore annotated by searching for amino-acidic regions that are members of known families: information derived from family architecture of a protein allows to infer its likely three-dimensional structure and function. Many protein family databases exist, and most of them use manually curated family definitions. These annotation methods (also known as annotation transfer by homology) become more relevant as the ratio between the number of structurally and functionally characterized proteins and the number of protein for which we only know the sequence decreases.

In this thesis we developed a method to automatically define protein families, based only on sequence data. In this way, we aimed to both i) exploit the homology information contained in large protein sequence databases, and ii) avoid definition biases that can be induced by human curation. This idea is of course not new. Indeed there are several algorithms in literature that tackle the problem, such as ADDA [7] and EVEREST [8]. The method presented here builds on these tools, with a particular attention to scalability with respect to the size of protein databases in the foreseeable future. It is an attempt to port in the field of protein annotation Density Peak Clustering [9], a recently developed algorithm which has been successfully used for unsupervised classification in many different fields.

The development of the algorithm, named DPCfam, has been the main task performed in my PhD. DPCfam's general workflow is described in Section 2.2, and it is based on a two-step clustering procedure performed with an automatized version of Density Peak Clustering. The procedure aims to first identify on single proteins those regions that are often found to be homologous with other proteins' regions; then, it groups together regions across different proteins that fall into putative families. In doing so, DPCfam uses solely local pairwise alignments data obtained from a large and non-redundant database (UniRef50 [10]). As shown in Section 3.2.4, the method is robust with respect to the few meta-parameters used, most of which have been selected on the basis of already known general criteria.

A major step in our work consisted in writing an efficient implementation of DPCfam. A first working implementation, named DPCfam0, is described in Section 2.3. This is a "developer" version of DPCfam, implemented for testing the method on two Pfam clans (PUA and P53-like) in the proof-of-principle experiment described in Section 3.2. As such, it is not intended to manage large quantities of alignments and can be used only for the clustering of small sequence datasets. A more mature implementation of DPCfam, parallelized and optimized, is described in Section 2.4. Its development required a major effort. Thanks to it, it has been possible to do a complete analysis of the UniRef50 database, which contains about 23 millions of sequences. To perform the various steps required by the method, we exploited as much as possible the intrinsic properties of its pipeline to accelerate computation without affecting, at any level, the quality of the results (both DPCfam implementations we developed will produce the same metaclusters if applied to the same dataset). This last implementation needs HPC facilities to run, and it has been developed with the help of a HPC expert. It is structured in a way that allows for incremental updates of metaclusters: when adding new sequences to the database it is not necessary to compute from scratch of the distance matrix (see Section 2.4.3), namely the most demanding step of the protocol, but it can be updated, thus dramatically reducing running times.

Results both for the proof-of-concept experiment (Section 3.2) and the complete UniRef50 analysis (Section 3.3) are encouraging. To assess the quality of metaclusters found by DPCfam, we decided to compare our results to Pfam's [2] families. This required another step in our work, namely defining a set of comparison measures, which are described in Section 3.1.1. Results obtained from the proof-of-concept experiment show a good agreement between Pfam and DPCfam's classifications. On one hand, DPCfam has been able to find, with similar boundaries, Pfam families that were well represented in the protein dataset. On the other hand, some of DPCfam's MCs show inconsistencies with respect to the Pfam classification. In some cases the MC's quality is actually low: such a poor quality can be assessed from some intrinsic property of the MC, such as the fraction of Low Complexity regions of its sequences, together with the distribution of the sequence's length (measures described in Section 3.1). In other cases, thanks to a in-depth analysis often performed using available structural data, we propose some possible adjustments in Pfam annotation, and we find a metacluster that may represent a novel protein family (see for example Section 3.2.2.1). An interesting feature of DPCfam arises from these experiments: it is possible to use it on small sequence datasets with meaningful results, and this is due to the fact that the protocol clusters local pairwise alignments between the small dataset and a large database (UniRef50).

The all-to-all clustering of the full UniRef50 database with DPCfam produced about 40,000 metaclusters. In this situation a direct comparison with the Pfam classification was possible. DPCfam finds about 30% of all Pfam families with a good boundary agreement: MCs of this type are named "equivalent". MCs can also represent smaller sub-regions of Pfam families: these are "reduced" MCs. These MCs can be the result of DPCfam splitting single domains into smaller, highly conserved, subregions or, on the contrary, of Pfam families covering multidomain regions, which are then broken up by DPCfam. It should also be stressed that MCs can be redundant, with different MCs covering overlapping parts of the same proteins. Additional MC categories are "extended" - MCs that cover regions larger than those of the Pfam family they map to - and "shifted" - MCs that typically have imperfect overlaps with the matching Pfam families. Taking all these cases into account, a total of 8,988 Pfam families map at least partially to one or more MCs. A very large number of MCs ($\sim 13,000$) clustered together regions with no Pfam annotation ("unknown" MCs): these constitute a very interesting set of MCs, defining possibly novel families. Understanding exactly what these MCs are is a major challenge for future analysis.

We analyzed in detail the fraction of MCs that exhibit at least some overlap with the Pfam classification, by looking at their evolutionary consistency, their sequence coverage and at the statistical properties of the profile-HMM that we derive from them. This was important to establish the ability of DPCfam to generate *bona fide* single or multi-domain families. In particular, we inspected the relationship between Pfam's profile-HMMs and DPCfam's profile-HMMs using "equivalent" MCs: in this case a direct comparison between the two models is indeed possible. We noted a strong agreement in terms of the hits respectively found by Pfam families and their "equivalent" models, suggesting they are actually representing the same family. While more work remains to be done on these MCs that might reveal interesting differences with the family Pfam classification (as seen in a few examples from the analysis of the proof-of-concept experiment in Section 3.2), likely the most important contribution of our work to protein family classification lies in the subset of "unknown" MCs, for which no member region could be found to have a Pfam annotation. We note that hits statistics for the profile-HMMs of "unknown" MCs are similar to those observed for those that map to known Pfam families, indicating that these are "sensible" models. The most important question regarding this subset of MCs, however, is what type of families they represent and in particular what structure and function they might have. While it is clear that a detailed answer to these questions will require a lot of manual work on individual families, a coarse-grained level, automatic analysis could still give us important hints on their nature. First, in order to verify whether a "low-hanging fruit" structural annotation might be available, we plan to run our Pfam-unannotated HMMs against sequences in the Protein Data Bank. This might allow identifying a number of MCs as known structural domains not yet classified by Pfam. Second, for those MCs that will return no significant match against PDB proteins [38] (likely the majority), we intend to take advantage of methods that predict generic structural features such as intrinsic disorder, transmembrane helices and coiled-coil regions. This would allow us to gain a better understanding of the structural universe that these MCs are sampling. Further, we plan to look into the phylogenetic classification of the organisms represented in each MC. Once again, this may reveal interesting features of our potential new families, this time in terms of the phylogenetic niches they occupy. Finally, we might want to run our MC-derived profile-HMMs against Pfam family and PDB protein profiles, using a method such as HHpred [44] for profile-profile alignment. Significant matches to Pfam families obtained in this way might suggest ways to expand existing (or create new) Pfam clans, while matches to PDB profiles will again provide structural and/or functional information potentially independently on the Pfam classification.

Besides gaining a deeper understanding of DPCfam's results obtained so far, there are future perspectives regarding the protocol itself. As we have discussed, protein families defined by DPCfam can overlap, showing a certain degree of redundancy mostly due to the possibility that a single MC captures a multi-family architecture, or a set of MCs finds different conserved subregions of a family (either overlapping or non-overlapping). As a further step in the protocol, it may then be necessary to implement a procedure to annotate proteins with non-overlapping DPCfam's families, either following strategies already adopted by other databases (e.g. Pfam's model-specific GAs) and/or through a reduction of MCs to "minimal" families, exploiting once again alignment information.

There is no doubt that the protocol can be improved in a number of other areas, such as the MCs' merging procedure (see Section 2.2.2.3). It is, in principle, possible to use pairwise alignments derived from different algorithms than BLAST, which can notably accelerate the entire clustering process. However, different alignment methods may result in different MCs' definitions, and the impact of such difference should be assessed before moving to faster methods. We also note that, while well optimized, the parallel implementation of DPCfam can be further improved in some of its steps.

As we already said, it is possible to recursively update DPCfam's MCs without re-reunning all the clustering steps. This is mostly doable thanks to the "block" strategy described in Section 2.4.3: we can add new blocks to the distance matrix, derived from new cluster groups obtained from new queries, and limit the calculations to such blocks instead of recomputing the whole matrix. The following step, the metaclustering procedure, is fast (see Section 2.4.4) and scales linearly with respect to the number of primary clusters. This will allow us, in the future, to offer to the scientific community updated versions of DPCfam as a database of MSAs or profile-HMMs or, given proper resources, as an interactive website.

Acknowledgements

A major part of this thesis couldn't be possible without the precious contribution of Federico Barone, HPC expert who wrote the parallel and optimized C++ code of DPCfam. This implementation is the result of his expertise together with long discussions we had on the subject. In understanding how and where we could best fasten the implementation, a precious contribution came from Dr. Francesca Rizzato and Dr. Alex Rodriguez. In addition, Federico Barone gave a major help in the analysis of the all-to-all clustering results, in particular on the Pfam coverage analysis. Thanks also to Dr. Alessio Ansuini and Dr. Stefano Cozzini, who allowed us to use the ORFEO HPC facility at Area Science Park and helped with technical support and precious discussion.

The supervision of Alessandro Laio and Marco Punta has been a *sine qua non.* I strongly believe that a better supervision could not be possible: both of them have been extremely involved in the whole work, teaching me that curiosity can not (and should not) decrease with expertise.

I need to thank all the SBP group, from which I always have been able to find help when needed: especially from Elena Facco, my "senpai", Andrea Papale, "The president", Andrea Saltalamacchia (the bio-man), Andrea Cesari, Giulia Sormani, Diego Doimo, Aldo Glielmo, Giulia Amici and Nicola Calonaci.

I would also thank Professor Pagnani and Professor Rosa for the help they gave me in some side-projects here not mentioned.

Being part of the Sissa community has been a nice journey. Time spent in Sissa has not always involved research: I want to thank the Director Ruffo and the Sissa Medialab for the Sissa4Schools and Students Day projects, which allowed me to do scientific communication activities. I need to cite again Francesca Rizzato, who involved me in these projects, beside others.

I also need to thank the Technology Transfer Office, especially Silvia Faion and Rene Butto, for a long list of occasions they gave me to apply academic knowledge outside academia. Thanks Rene for the long talks we had: not always in agreement, but very fruitful.

Thank you to all the other student representatives who shared with me the journey in the Student's Council with extremely high levels of passion: The Presidents, Andrea Papale and Alessandro Nobile, Gabriele Perfetto who sat with me at the Board of Directors, Manuela Santo and Mara De Rosa, Alessandro Rubin, among the others. Also thanks to all the ADI (Associazione Italiana Dottori e Dottorandi di Ricerca in Italia) members, in Trieste and in Italy.

Thanks to the old friends (Dax, Irene, Anna, Teo, Bianca, Edo, Arianna and Pietro, Francesco, Erica, Lorenzo), the oldest friends (Giulia, Anna, i Ludri), the new friends (Martina, Andrea, Francesca).

Thanks to my parents, my brother and my Pina, who all always supported me.

Going back to page (i): no, no one is the only maker of his/her fate.

Bibliography

[1] A. Scaiewicz and M. Levitt, “The language of the protein universe,” *Current opinion in genetics and development*, vol. 35, pp. 50–56, 2015.

[2] S. El-Gebali *et al.*, “The Pfam protein families database in 2019,” *NAR*, vol. 47, pp. D427–D432, 10 2018.

[3] J. Gough *et al.*, “Assignment of homology to genome sequences using a library of hidden markov models that represent all proteins of known structure11edited by g. von heijne,” *JMB*, vol. 313, no. 4, pp. 903 – 919, 2001.

[4] T. E. Lewis *et al.*, “Gene3D: Extensive prediction of globular domains in proteins,” *NAR*, vol. 46, pp. D435–D439, 11 2017.

[5] H. Cheng *et al.*, “Manual classification strategies in the ECOD database,” *Proteins*, vol. 83, pp. 1238–1251, Jul 2015.

[6] A. L. Mitchell *et al.*, “InterPro in 2019: improving coverage, classification and access to protein sequence annotations,” *NAR*, vol. 47, pp. D351–D360, 11 2018.

[7] A. Heger and L. Holm, “Exhaustive enumeration of protein domain families,” *JMB*, vol. 328, no. 3, pp. 749 – 767, 2003.

[8] E. Portugaly *et al.*, “Everest: automatic identification and classification of protein domains in all protein sequences,” *BMC bioinformatics*, vol. 7, no. 1, p. 277, 2006.

[9] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[10] TheUniProtConsortium, “Uniprot: the universal protein knowledgebase,” *NAR*, vol. 45, no. D1, pp. D158–D169, 2017.

[11] R. Ishitani *et al.*, “Crystal structure of archaeosine trna-guanine transglycosylase,” *JMB*, vol. 318, no. 3, pp. 665 – 677, 2002.

[12] M. Berardi *et al.*, “The ig fold of the core binding factor alpha runt domain is a member of a family of structurally and functionally related ig-fold dna-binding domains,” *Structure (London, England : 1993)*, vol. 7, p. 1247—1256, October 1999.

[13] M. Cobb, “60 years ago, francis crick changed the logic of biology,” *PLoS biology*, vol. 15, no. 9, p. e2003243, 2017.

[14] G. A. Petsko, “Dog eat dogma,” *Genome Biology*, vol. 1, no. 2, pp. comment1002–1, 2000.

[15] C. P. Ponting and R. R. Russell, “The natural history of protein domains,” *Annual review of biophysics and biomolecular structure*, vol. 31, no. 1, pp. 45–71, 2002.

[16] J. Liu and B. Rost, “Domains, motifs and clusters in the protein universe,” *Current opinion in chemical biology*, vol. 7, p. 5—11, February 2003.

[17] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt, “Direct-coupling analysis of residue coevolution captures native contacts across many protein families,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 49, pp. E1293–E1301, 2011.

[18] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[19] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks,” *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10915–10919, 1992.

[20] M. Dayhoff, R. Schwartz, and B. Orcutt, “A model of evolutionary change in proteins,” *Atlas of protein sequence and structure*, vol. 5, pp. 345–352, 1978.

[21] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis: probabilistic models of proteins and nucleic acids*, ch. 2 Pairwise Alignments. Cambridge university press, 1998.

[22] T. F. Smith, M. S. Waterman, *et al.*, “Identification of common molecular subsequences,” *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[23] I. Korf, M. Yandell, and J. Bedell, *Sequence Similarity*, ch. 4. ” O’Reilly Media, Inc.”, 2003.

[24] 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*, vol. 87, no. 6, pp. 2264–2268, 1990.

[25] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *Journal of molecular biology*, vol. 215, no. 3, pp. 403–410, 1990.

[26] C. Camacho *et al.*, “BLAST+: architecture and applications,” *BMC Bioinformatics*, vol. 10, p. 421, Dec 2009.

[27] C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. L. Madden, “Blast+: architecture and applications,” *BMC bioinformatics*, vol. 10, no. 1, p. 421, 2009.

[28] I. Korf, M. Yandell, and J. Bedell, *Multiple Sequence Alignments*, ch. 6. ” O’Reilly Media, Inc.”, 2003.

[29] C. Notredame, D. G. Higgins, and J. Heringa, “T-coffee: A novel method for fast and accurate multiple sequence alignment,” *Journal of molecular biology*, vol. 302, no. 1, pp. 205–217, 2000.

[30] F. Sievers and D. G. Higgins, “Clustal omega,” *Current protocols in bioinformatics*, vol. 48, no. 1, pp. 3–13, 2014.

[31] R. C. Edgar, “MUSCLE: multiple sequence alignment with high accuracy and high throughput,” *NAR*, vol. 32, pp. 1792–1797, 03 2004.

[32] S. R. Eddy, “Bioinformatics review profile hidden markov models,” *Bioinformatics*, vol. 14, pp. 755–763, 1998.

[33] S. R. Eddy, “Hidden markov models,” *Current Opinion in Structural Biology*, vol. 6, no. 3, pp. 361 – 365, 1996.

[34] J. Mistry *et al.*, “Challenges in homology search: HMMER3 and convergent evolution of coiled-coil regions,” *NAR*, vol. 41, pp. e121–e121, 04 2013.

[35] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Markov chains and hidden Markov models*, ch. 3. Cambridge university press, 1998.

[36] B. Boeckmann *et al.*, "The swiss-prot protein knowledgebase and its supplement trembl in 2003," *NAR*, vol. 31, no. 1, pp. 365–370, 2003.

[37] W. Li and A. Godzik, "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.

[38] wwPDB consortium, "Protein data bank: the single global archive for 3d macromolecular structure data," *Nucleic acids research*, vol. 47, no. D1, pp. D520–D528, 2019.

[39] A. G. Murzin *et al.*, "Scop: a structural classification of proteins database for the investigation of sequences and structures," *JMB*, vol. 247, no. 4, pp. 536–540, 1995.

[40] E. L. Sonnhammer, S. R. Eddy, and R. Durbin, "Pfam: a comprehensive database of protein domain families based on seed alignments," *Proteins: Structure, Function, and Bioinformatics*, vol. 28, no. 3, pp. 405–420, 1997.

[41] E. L. Sonnhammer and D. Kahn, "Modular arrangement of proteins as inferred from analysis of homology," *Protein Science*, vol. 3, no. 3, pp. 482–492, 1994.

[42] R. D. Finn *et al.*, "The pfam protein families database: towards a more sustainable future," *NAR*, vol. 44, no. D1, pp. D279–D285, 2016.

[43] M. Hauser, M. Steinegger, and J. Söding, "Mmseqs software suite for fast and deep clustering and searching of large protein sequence sets," *Bioinformatics (Oxford, England)*, vol. 32, p. 1323—1330, May 2016.

[44] L. Zimmermann *et al.*, "A completely reimplemented mpi bioinformatics toolkit with a new hhpred server at its core," *JMB*, vol. 430, no. 15, pp. 2237 – 2243, 2018. Computation Resources for Molecular Biology.

[45] F. Corpet, J. Gouzy, and D. Kahn, "The prodom database of protein domain families," *Nucleic Acids Research*, vol. 26, no. 1, pp. 323–326, 1998.

[46] E. L. Sonnhammer and D. Kahn, "Modular arrangement of proteins as inferred from analysis of homology," *Protein Science*, vol. 3, no. 3, pp. 482–492, 1994.

[47] A. J. Enright *et al.*, "An efficient algorithm for large-scale detection of protein families.," *NAR*, vol. 30 7, pp. 1575–84, 2002.

[48] R. Sokal and C. Michener, "A statistical method for evaluating systematic relationships," *UNIVERSITY OF KANSAS SCIENTIFIC BULLETIN*, 1958.

[49] M. P. Forum, "Mpi: A message-passing interface standard," 1994.

[50] I. J. Davis, "A fast radix sort," *The computer journal*, vol. 35, no. 6, pp. 636–642, 1992.

[51] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces.* Arpaci-Dusseau Books, 1.00 ed., August 2018.

[52] L. Holm, "Benchmarking fold detection by DaliLite v.5," *Bioinformatics*, vol. 35, pp. 5326–5327, 07 2019.

[53] P. Mier, L. Paladin, S. Tamana, S. Petrosian, B. Hajdu-Soltész, A. Urbanek, A. Gruca, D. Plewczynski, M. Grynberg, P. Bernadó, Z. Gáspári, C. A. Ouzounis, V. J. Promponas, A. V. Kajava, J. M. Hancock, S. C. E. Tosatto, Z. Dosztanyi, and M. A. Andrade-Navarro, "Disentangling the complexity of low complexity proteins," *Briefings in Bioinformatics*, vol. 21, pp. 458–472, 01 2019.

[54] S. R. Eddy, "A new generation of homology search tools based on probabilistic inference," in *Genome Informatics 2009: Genome Informatics Series Vol. 23*, pp. 205–211, World Scientific, 2009.

[55] L. M. Iyer *et al.*, "The ASCH superfamily: novel domains with a fold related to the PUA domain and a potential role in RNA metabolism," *Bioinformatics*, vol. 22, pp. 257–263, 12 2005.

[56] S. Jones *et al.*, "Domain assignment for protein structures using a consensus approach: Characterization and analysis," *Protein Science*, vol. 7, no. 2, pp. 233–242, 1998.

[57] J. Qian, N. M. Luscombe, and M. Gerstein, "Protein family and fold occurrence in genomes: power-law behaviour and evolutionary model," *Journal of molecular biology*, vol. 313, no. 4, pp. 673–681, 2001.

[58] M. Punta *et al.*, "The pfam protein families database," *NAR*, vol. 40, no. D1, pp. D290–D301, 2012.

[59] H. d. t. Sean R. Eddy, Travis J. Wheeler, "Hmmer user's guide," 2015. `http://eddylab.org/software/hmmer3/3.1b2/Userguide.pdf`.

[60] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.