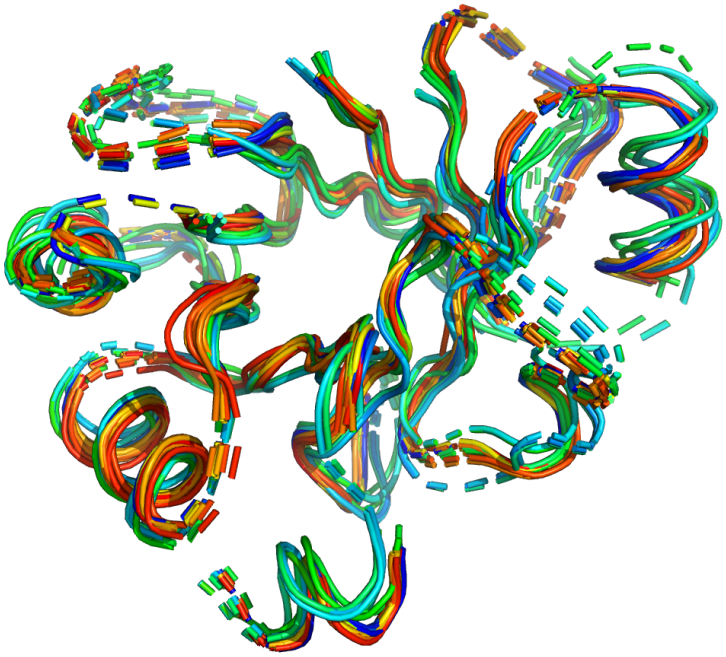
Modeling and Characterization of Nitronate Monooxygenase (PF03060) Domain

Biological Data Project

Gianmarco Cracco, Alessandro Manente,
Riccardo Mazzieri

16th February 2021



Rendering of PDB structures common core.

Contents

Introduction

The aim of this project was to conduct an in-depth study on a specific protein domain, namely the Nitronate Monooxygenase domain (PFAM ID: PF03060).

All the produced code can be found on GitHub[1], and it can be executed in its entirety through the Jupyter Notebook and the use of some bash scripts (which require a Linux System to be run). All the details can be found in the `README.md` file. Finally, in several steps of the analysis we downloaded the needed data or databases from online resources like UniProt or PDB: we will report in detail each of these steps so that the analysis is fully reproducible.

[1] https://github.com/Nihmar/BiologicalDataDS2020

1 Sequence Model

The first task of the project was to build a sequence model for our domain family. The first step was to retrieve the ground truth data, meaning all the proteins in Swiss-Prot database that are annotated with our Pfam domain . To do this, we used the InterPro APIs which we accessed from code, and we found that 23 sequences in the Swiss-Prot database are annotated with our Pfam family.

1.1 Model Creation

After getting the ground truth data (we call that GT), we were ready to build our models. Our first task was to retrieve homologous sequences starting from our domain sequence. To do so, we performed database queries both against Swiss-Prot and UniRef90, and considering only the significant hits (E-value < 0.01). We made several attempts of database queries, and we report here only some of them. After a first attempt using SwissProt, we tried to obtain more sequences by making a BLAST search on the UniProt web service[2], searching against Uniref90, and accepting also non-annotated proteins. We obtained of course a much larger amount of hits, together with much smaller E-values, the larger being around 2e-42. In our tests, we considered both the first 250 and 100 hits to generate our models. For later use, we label the resulting sets of sequences deriving from the described BLAST searches as 1, 2 and 3 respectively.

For each of those sets of retrieved hits, we performed a multiple sequence alignment (MSA). To do this, we used three different algorithms: T-Coffee, Muscle and Clustal-Omega. All the algorithms were run on the EMBL-EBI web service[3] and also in this case we used the default parameters. For some of those MSAs we also removed the non conserved columns at the beginning and the end of the alignment since this could be a source of noise for our model.
For each of those MSAs, we built a PSSM and a HMM, then performed a PSI-BLAST search and HMM-Search against Swiss-Prot, using respectively our PSSM and our HMM as inputs. To do all of this we used the command line versions of `blast+` and `hmmer`, and automated all the process with the help of some simple bash scripts. For each PSI-BLAST search, we also tried to make a different amount of iterations each time. We will denote each model with an abbreviation of the form XYW(d), where:

- X = The type of model ((P)SSM, (H)MM);
- Y = MSA algorithm used (T-(C)offee, (M)USCLE, Clustal-(O)mega);
- W = The sequences used to build the MSA;
- d = Present if the MSA was denoised, not present otherwise.

For example, *PC1d* will denote the PSSM model built from the denoised MSA generated from the set of sequences 1 and with the T-Coffee algorithm. Since we created and tested

[2]https://www.uniprot.org/blast/
[3]https://www.ebi.ac.uk/Tools/msa/

a total of 32 models, we report here just the top 10 models in terms of performance. We will discuss later how we decided what metrics to use to assess model quality.
All the results we will discuss here were computed considering only the hits from the models with E-value < 0.001.

1.1.1 Evaluating ability of matching sequences

At this point, in order to evaluate the ability of our model to find the right sequences, we counted how many sequences our models:

- correctly predicted to contain the domain (TP);
- wrongly predicted to contain the domain (FP);
- correctly predicted not to contain the domain (TN);
- wrongly predicted not to contain the domain (FN).

Once those values were computed, we made a summary table of all the useful statistics that in our case better represented the ability of the model to find the correct sequences, which can be found in Table 1. To compare them, we incorporated the statistics into a handful of relevant scores: precision, recall, MCC (Matthews Correlation Coefficient) and F1-Score. We also computed accuracy and balanced accuracy for each model, but since the dataset we are working with is extremely unbalanced (23 positive cases vs. the rest of the entire Swiss-Prot database of negative cases), those weren't at all useful to evaluate model performances and we didn't consider them.
The models presenting "1_it" at the end mean that we evaluated the results after just one iteration of PSI-BLAST. Highlighted in red is the best performing model for ease of reading. More of this on Sec. 1.2

	n_hits	Precision	Recall	MCC	F1-score
PM3d1_it	51	0.431	0.957	0.642	0.595
PM31_it	53	0.415	0.957	0.630	0.579
PM21_it	67	0.328	0.957	0.560	0.489
PO1	79	0.278	0.957	0.516	0.431
HM3d	79	0.278	0.957	0.516	0.431
HC3d	81	0.272	0.957	0.510	0.423
HM3	82	0.268	0.957	0.507	0.419
HO3d	82	0.268	0.957	0.507	0.419
HC3	83	0.265	0.957	0.503	0.415
HO3	86	0.256	0.957	0.495	0.404

Table 1: Performances of the best performing models.

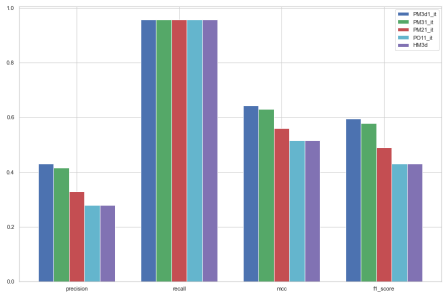


Figure 1: Bar plot of 5 best performing models.

1.1.2 Evaluating the Ability of Matching Domain Positions

At this point, we also wanted to evaluate the ability of our model to find the domain positions inside the found proteins.

To do this, we computed an analogous of the statistics and scores mentioned above, so True Positives, False Positives, True Negatives and False Negatives, and then precision, recall, MCC and F1-Score.

Note that, technically, we should consider as True Negatives also all the proteins in Swiss-Prot that our model correctly predicted not to have the domain (that is, all the proteins that our model didn't find, and that actually don't have the domain). While this is true, doing so any model would find an extremely high number of TN[4]. This statistic would not help us in any way to assess the ability of our models to find domain positions, and would significantly slow down our computation times. For this reason we decided to exclude this quantity from our analysis. We report our results with some of the best performing models in Table 2.

	Precision	Recall	MCC	F1-score
PM3d1_it	0.722	0.949	0.701	0.820
PM31_it	0.696	0.955	0.685	0.805
PM21_it	0.639	0.950	0.662	0.764
PO21_it	0.520	0.963	0.624	0.675
PC2d1_it	0.486	0.956	0.606	0.645
HM3d	0.466	0.940	0.602	0.624
PC3d1_it	0.465	0.955	0.578	0.626
PO3d1_it	0.465	0.957	0.576	0.626
PC31_it	0.464	0.961	0.576	0.626
HC3d	0.460	0.952	0.602	0.620

Table 2: Performances of the best performing models at predicting domain positions.

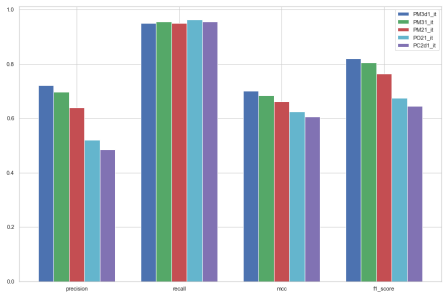


Figure 2: Bar plot of 5 best performing models.

1.2 Choosing the best model

After computing relevant statistics, we had to decide which was the best model to pick. The main thing we noticed was that every model we built had extremely high recall, but low precision. This meant that our model didn't struggle to find the proteins containing annotations of our Pfam domain, but also ended up finding a lot more proteins that didn't contain it. Also, all the hits had significantly low E-values, well below 0.001. We decided therefore that the most important metric for our purposes was precision, also keeping an eye on MCC and F-Score, which are good scores for unbalanced situations like this as well. Sorting the results by precision we found that our best model was **PM3d1_it**. We used the proteins found by this model to continue our analysis about structural and functional characterization.

[4]Specifically, the sum of all the lengths of all the proteins in Swiss-Prot that were not found by our model, and that are not present in GT.

2 Domain family characterization

2.1 Dataset definitions

The second part of the analysis is based on two datasets, that we built from the 88 proteins found by the PM3d1_it model (which we will, from now on, refer to as *best model*). We will also refer to this set of proteins as the *model proteins*.n

- **Family structures database**: the set of known structures among our model proteins. To retrieve this data, we used the `uniprot_segments_observed.tsv` mapping database downloaded from the SIFTS service[5], which allowed us to retrieve the PDB IDs of such structures. Once we got the IDs, we downloaded the relative PDB files from the PDB website[6].
- **Family sequences database**: the set of model proteins sequences, clustered by 90% sequence identity. To obtain the UniProt IDs associated to such sequences, we mapped UniProt IDs of the model proteins to UniRef90 using the retrieve/ID Mapping from UniProt[7].

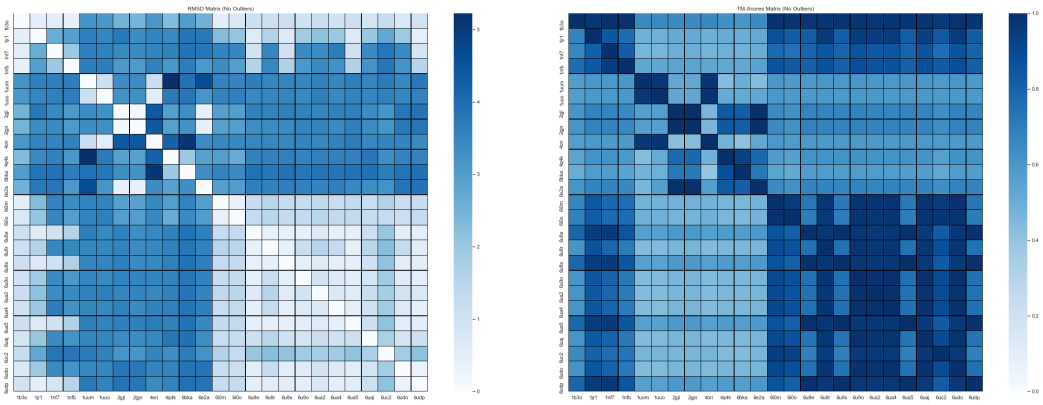
2.2 Structural characterization

In this part of the analysis we considered the family structures database, to study the structural features of our retrieved proteins.

2.2.1 Pairwise RMSD and TM-Score

The first thing we did was to perform a pairwise all-vs-all multiple sequence alignment, using the TM-Align command line tool. We report the resulting pairwise RMSDs and TM-Scores in Fig. 3. We also produced an hierarchical clustering with respect to those scores, and reported them in the dendograms in Fig. 4 and 5, to see which groups were more similar between each other. Furthermore, we found, looking at the produced matrices and dendograms, the presence of a couple of outliers (PDB IDs `6u8n` and `6udq`) and removed them.
Here for sake of brevity, we reported only the final images, already without outliers, the original ones can be found in Sec. A in Fig. 12, 13 and 14.

[5] https://www.ebi.ac.uk/pdbe/docs/sifts/quick.html
[6] https://www.rcsb.org/downloads
[7] https://www.uniprot.org/uploadlists/



(a) (b)

Figure 3: (a) RMSD matrix (No Outliers), (b) TM-Score matrix (No Outliers).

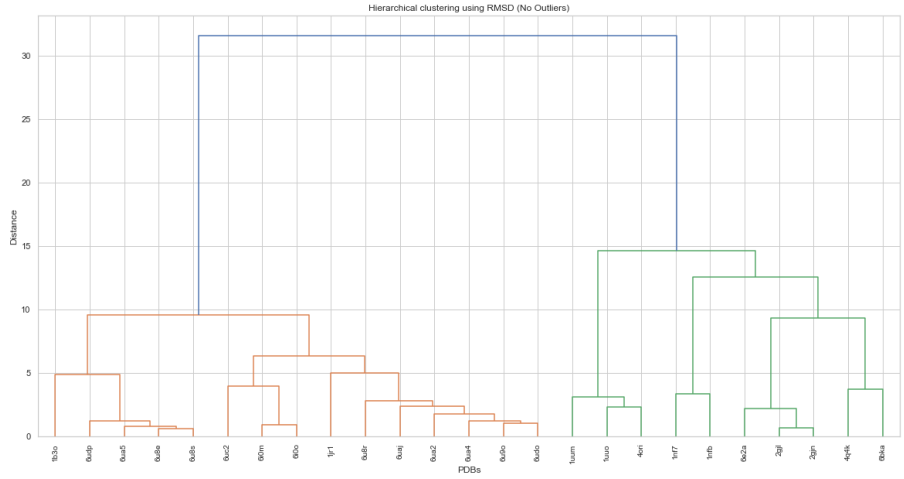


Figure 4: Dendogram based on RMSD without ouliers.

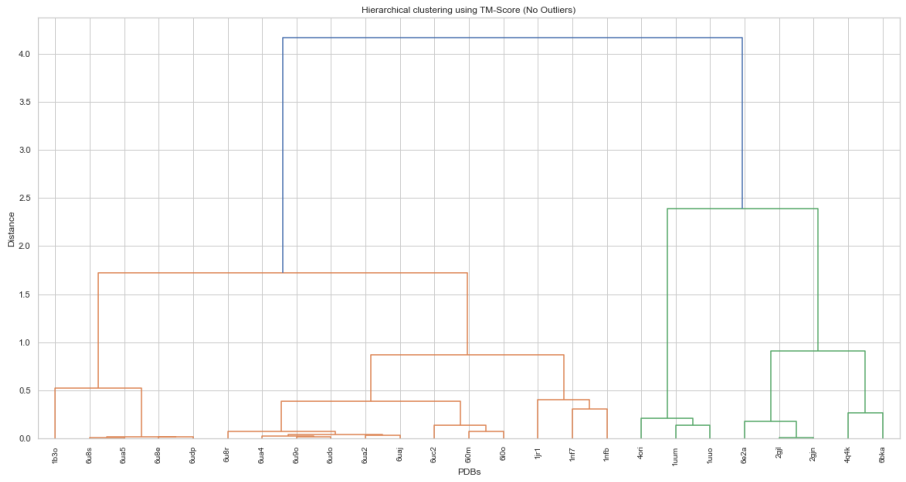


Figure 5: Dendogram based on TM-score without ouliers.

2.2.2 Multiple Structural Alignment and Conserved Contacts.

We then performed multiple structural alignment on our dataset to identify and visualize conserved positions with the mTM-Align web service[8].
To visualize the conserved positions we started considering the PDB files we obtained from the mTM-Align output, which contained the superimposed structures of our proteins, both for the complete structures and the common conserved core. We considered the latter, containing the common core structure, and for each of our PDB files we created distance matrices and contact maps, with a distance threshold between Carbon Alpha atoms of 8 Å. Then, in order to visualize only the conserved positions we created a single contact map which is the average of all the contact maps. We also created the same contact map, but with sequence separation equal to 12, to identify long range contacts. The results are visible in Fig. 6 and Fig. 7.
We also provided a visualization of the conserved regions thanks to the CM-View tool[9], the results of which are shown in Fig 8.
At the end we tried to make good use of the other mTM-Align output: the complete structure. To do so we plotted again distance matrices for each PDB, but this time considering both the entire structure and the common core; the result is visible in Sec. A, Fig. 15.

[8]https://yanglab.nankai.edu.cn/mTM-align/
[9]http://www.bioinformatics.org/cmview/index.html

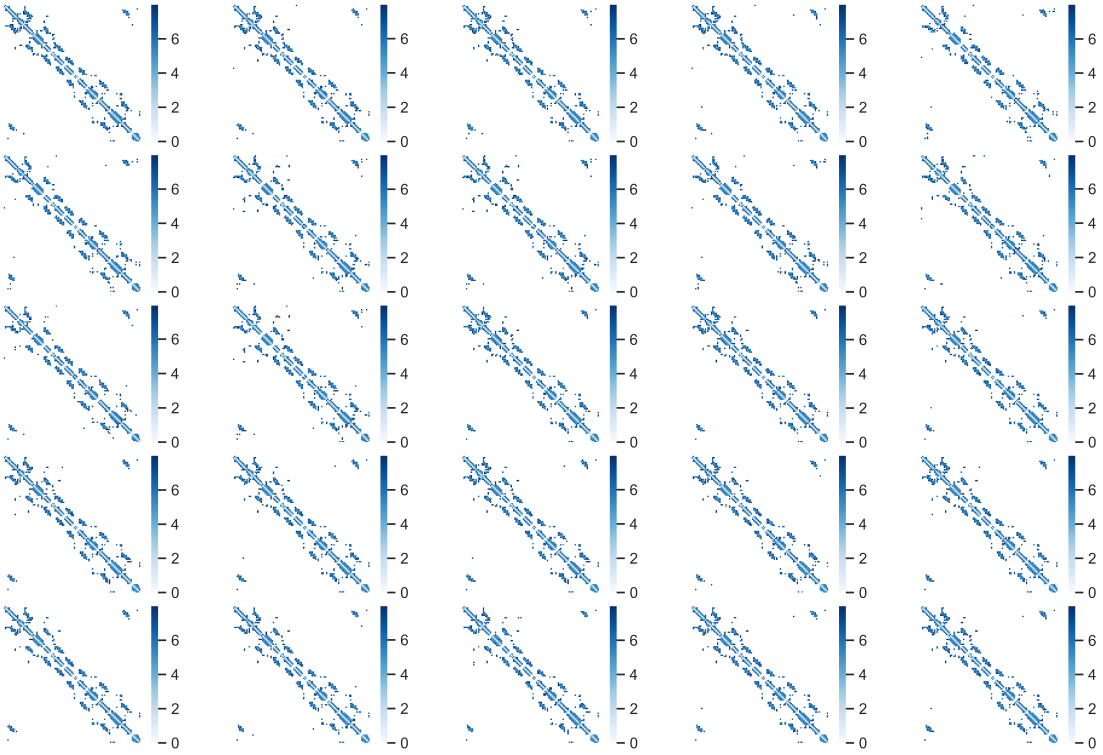
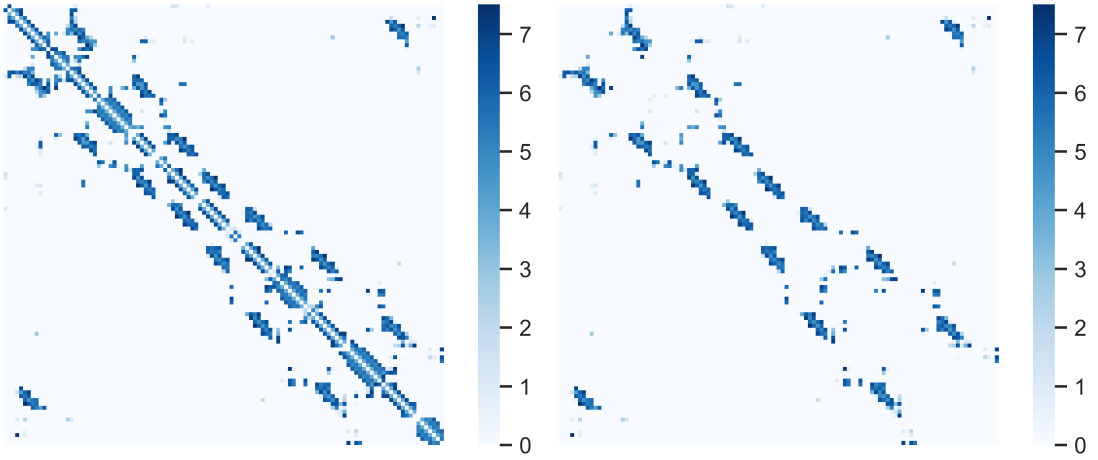
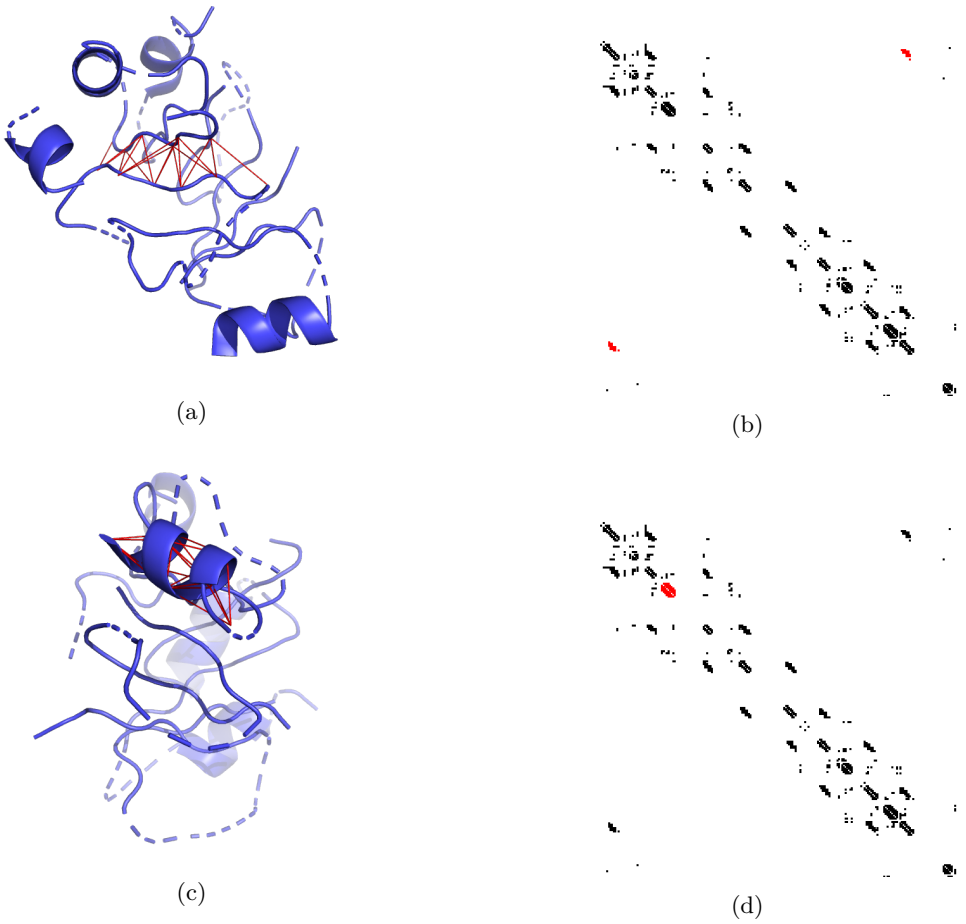


Figure 6: Contact maps for each of our PDB structures.



(a) (b)

Figure 7: (a) Average contact map and (b) average long range contact map, with sequence separation = 12.



(a) (b) (c) (d)

Figure 8: With CM-View, we managed to obtain a 3D visualization of the conserved common core of one of the structures of our dataset (PDB ID `1B30`), and its relative contact map. We used this tool to better interpret the meaning of the contact maps we obtained in our analysis. On the contact maps (b,d) we highlighted in red specific patterns, and, on the 3D view (a,c), the respective structures were highlighted by CM-View. We can see that the main conserved secondary structures are α-helices (close range contacts) and parallel β-sheets (long range contacts). Specifically, comparing these images with the contact maps we obtained in python, we can see that the entirety of the long range conserved contacts is made up of parallel β-sheets.

2.2.3 CATH Superfamily and Family

After studying the structural features of our domain, we had to determine to which CATH family and/or superfamily our protein domain belonged. To do this we used two distinct databases: the first, `pdb_chain_cath_uniprot.tsv`, contains mappings between Uniprot and PDB IDs to CATH IDs, and was downloaded from the SIFTS website. We used it to retrieve all the model proteins that belong to a CATH domain, and their

relative CATH ID. The second one was downloaded from the CATH website[10], and contains all the CATH domain/superfamily pairs, and we used it to map CATH IDs to CATH superfamilies. We found that, from the 88 proteins found by our models, only 21 map to a CATH domain, and all of those CATH domains belong to the `3.20.20.70` superfamily.

2.3 Taxonomy

The goal for this step was to obtain taxonomy lineage for each protein contained in the `family sequences database`, in order to plot the taxonomic tree. To do this, we searched against all UniProt using our family sequences dataset as a query: this was done to obtain exactly the same proteins as a result, and to be able to access to all the annotations available. Then, via the available column customization, we managed to obtain the taxonomic features, such as the taxonomic ID and the taxonomic lineage. We finally downloaded everything in `.tab` format, to easily read it from code.

To actually generate the taxonomic tree from the taxonomic lineage data, we used the Taxonomy Common Tree service available from the NCBI website[11]. Specifically, we extracted in a `.txt` file all the taxonomic IDs and used that as input for the NCBI service. Doing that, we managed to obtain a `.phy` file. Once a dictionary containing the frequencies of each Taxonomic ID was obtained, we managed to plot our taxonomic tree with its node size proportional to their relative abundance, which we show in Fig. 9.

[10] ftp://orengoftp.biochem.ucl.ac.uk/cath/releases/daily-release/newest/
[11] https://www.ncbi.nlm.nih.gov/Taxonomy/CommonTree/wwwcmt.cgi

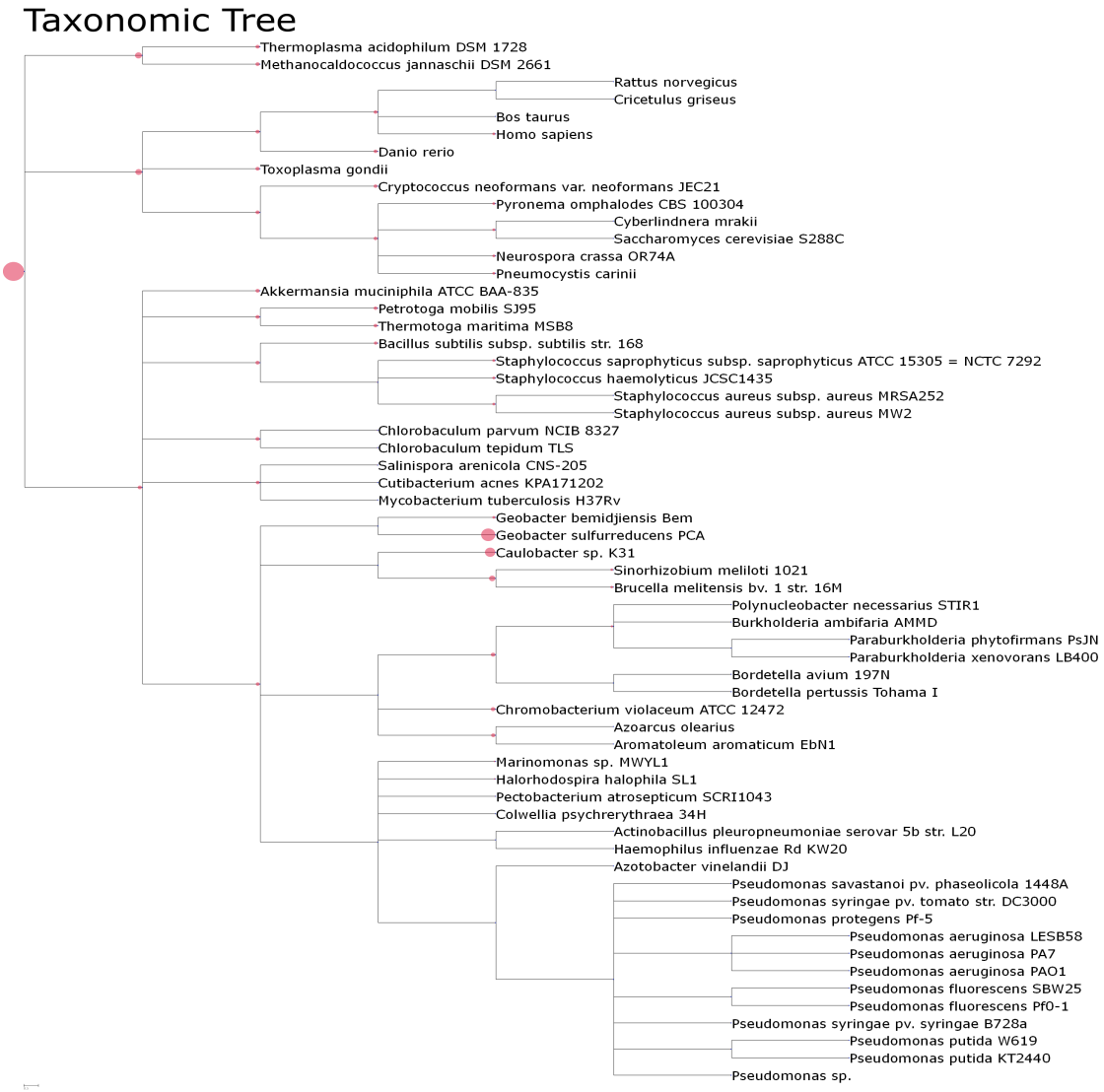


Figure 9: Taxonomic tree.

2.4 Functional characterization

At this point our focus was to give a functional characterization of our family sequences dataset. For this section we used two databases. The first is the `go.obo` database, which contains the whole Gene Ontology tree, which we downloaded from the Gene Ontology website[12]. For the annotations, we considered the whole set of Gene Ontology Annotations of Swiss-Prot, which we manually downloaded as a `.tab` file from UniProt using the advanced search and looking for Gene Ontology terms both manually and automatically annotated.

[12]http://geneontology.org/docs/download-ontology/

2.4.1 Enriched Terms

To find the enriched terms, we first computed the confusion tables (showed in Table 3) for each term present in our family sequences database. Starting from the tables, we then computed, for each GO term, both the fold increase and the left, right and two-tail p values from the exact Fisher test.

	Having the property	Not having the property
Selected	a	b
Not selected	c	d

Table 3: Confusion table for each GO Term. From this we can compute the fold increase, which is equal to $\frac{a}{b} \cdot \frac{d}{c}$.

As expected, we found that the terms with highest fold increase were also the ones with right and two-tail p-values close to 0. We visualized the most enriched terms in a word cloud, which can be found in Fig. 10. In the word cloud, the terms size is proportional to their fold increase.

Figure 10: Word cloud with the most enriched terms. The word color denotes the namespace they belong to: blue stands for Biological Process, purple stands for Molecular Function, and green stands for Cellular Component. Some terms were trimmed since they were too long and didn't allow for a pleasing visualization.

2.4.2 High Level Enriched Terms

In the previous section we found the most enriched terms. On a closer inspection we found out that the majority of our enriched terms were leaf terms. In this section we also find the high level enriched terms, i.e. the ones that have more children terms. So to try to see which high level terms were more enriched we filtered our data looking only for GO terms with a higher number of children with respect to the 80% percentile of all our enriched terms, and then sorted them by decreasing fold increase. We plot the most enriched terms, proportionally to their fold increase, on the word cloud in Figure 11.

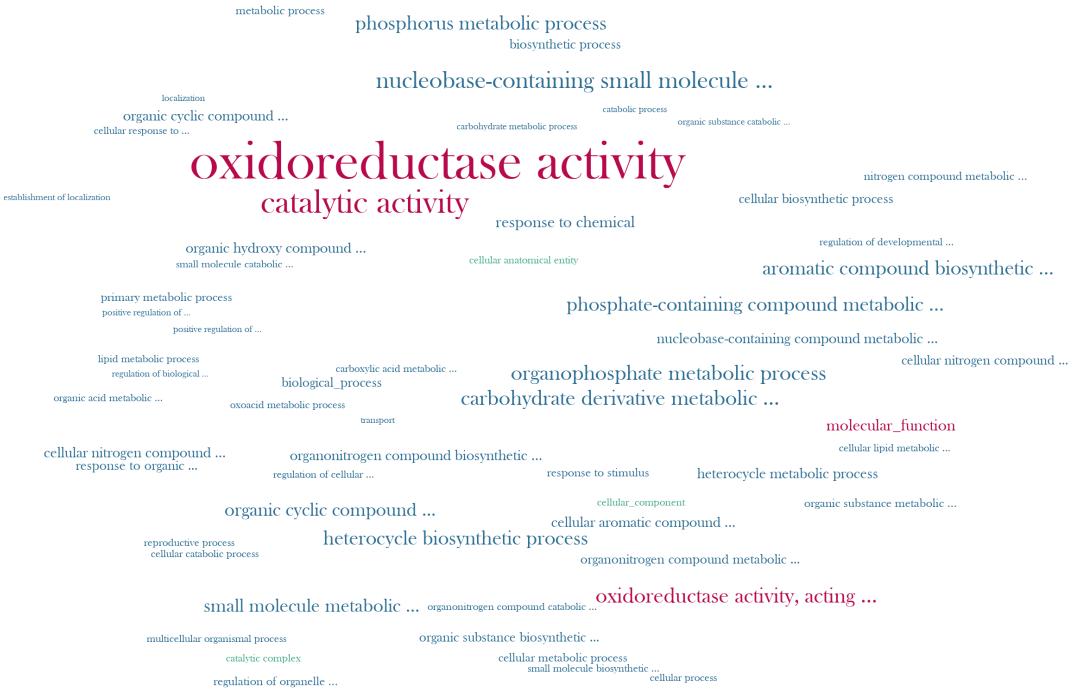
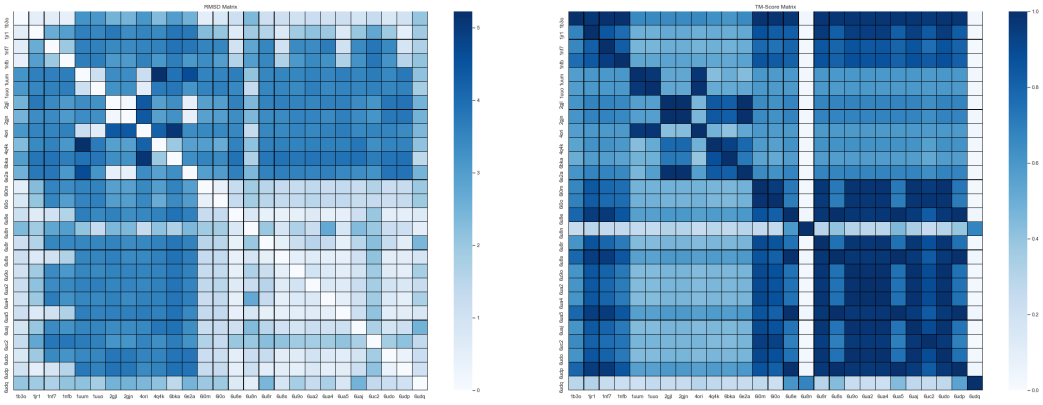


Figure 11: Word cloud for high level terms.

A Appendix: Additional Plots



(a) (b)

Figure 12: (a) RMSD matrix, (b) TM-Score matrix, both considering outliers.

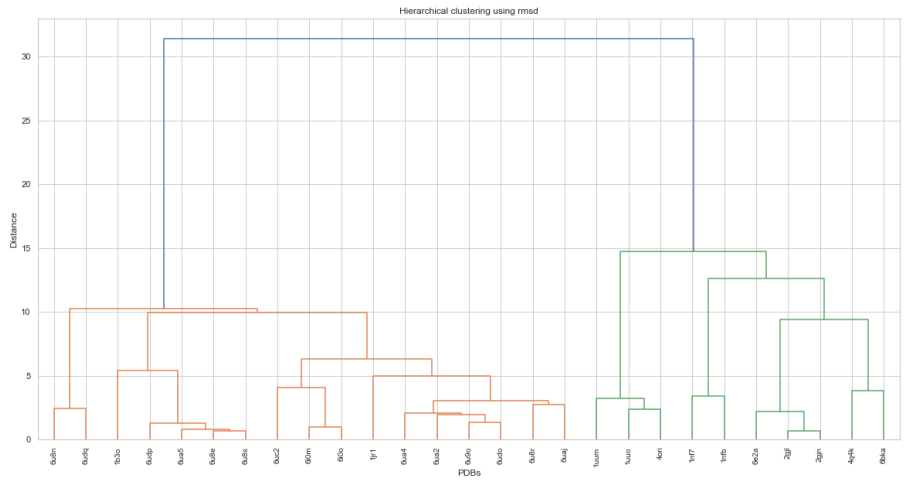


Figure 13: Dendogram based on RMSD, considering outliers.

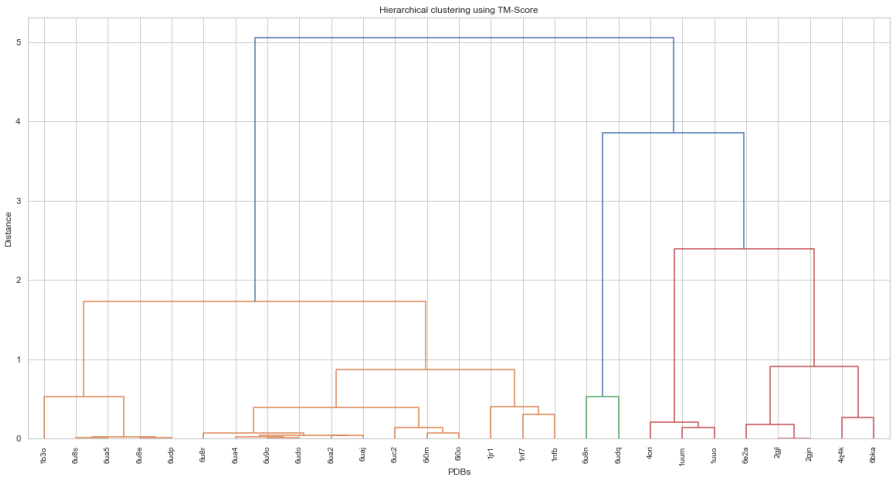


Figure 14: Dendogram based on TM-score, both considering outliers.

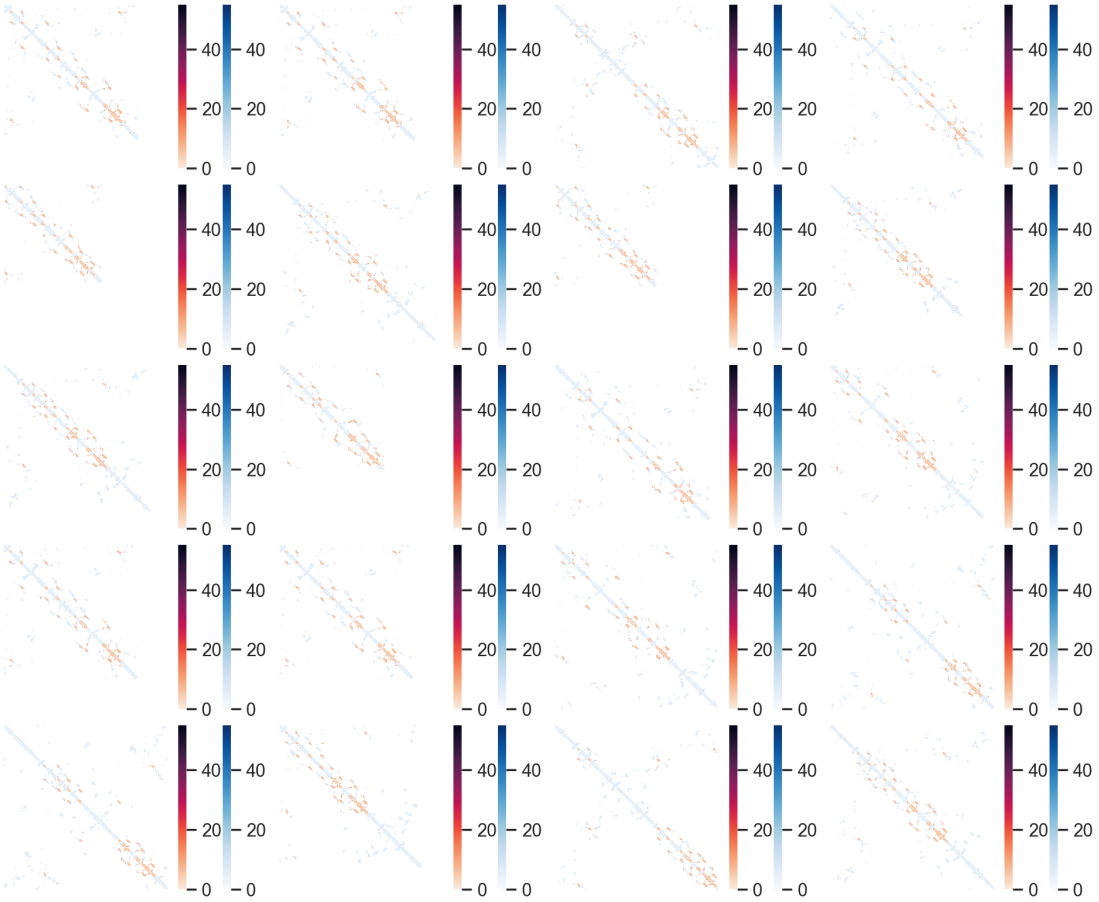


Figure 15: Plots of all the contact maps of the complete structures. Highlighted are the conserved common core regions.