Report
Biological Data Project
2020/2021

Group 14

Davide Ghiotto - 1236660
Matteo Lavina - 1227466
Darko Ivanovski - 1243085

Group data input

Each group received a representative sequence of the domain family.
We report here our assigned data for reference:

Group	UniProt	Organism	Pfam ID	Pfam name	Domain position	Domain sequence
14	Q01638	Homo sapiens (Human)	PF01582	Toll/interleukin-1 receptor homology (TIR) domain	379-541	AYVVYPRNYKSSTDGASR VEHFVHQILPDVLENKCGY TLCIYGRDMLPGEDVVTAV ETNIRKSRRHIFILTPQITHN KEFAYEQEVALHCALIQND AKVILIEMEALSELDMLQAE ALQDSLQHLMKVQGTIKW REDHIANKRSLNSKFWKH VRYQMPVPSKI

Model building

1. **Ground truth**
   In order to find the list of proteins annotated with the Pfam identifier we used InterPro API filtering for our Pfam ID and obtained a json with all the annotated proteins.
   We found 148 proteins annotated.
   The list of all the rest of the proteins in SwissProt that are not annotated with our Pfam ID were downloaded from the UniProt website.

   ```
   > script model/define_ground_truth.py
   ```

2. **Homologus proteins**
   We performed a BLAST search on UniProt using NCBI web service with standard parameters and the sequence was the one assigned to our group.
   We obtained 34 hits.

3. **Multiple Sequence Alignment**
   We generated a MSA using t-coffee web service with standard parameters starting from the BLAST search of the previous point.

4. **Remove noise**
   Using Jalview we visualized the alignment and removed the noise (non conserved positions) presented at the end of the alignment (for one test for the HMM model).

5. **PSSM model**
   After performing a PSI-BLAST search, at second iteration, we downloaded the model directly from the web interface.

6. **HMM model**
   We downloaded the HMM model from the Skylign web service starting from the MSA.

7. **Significant hits**
   a. **PSI-BLAST:** blast web service at ncbi
      i. 181 significant hits, starting from a model without removing noise
   b. **HMM-SEARCH:** we used hmmsearch web service uploading directly the HMM model of the previous step
      i. First version 132 hits
      ii. Second version 53 hits (HMM model built from MSA after removing noise)

8. **Evaluate matching ability for the sequences**
   In order to evaluate the matching ability of a model in terms of sequences we implemented a simple algorithm that allows us to classify our data in a Confusion Matrix. For every sequence in our model we checked if it was present in the ground truth: in this way we can define the 4 cases of the standard classification problem (TP, TN, FP, FN).

After this computation we extracted the following useful statistics: balanced accuracy, f1, MCC. Given the structure of the problem, it would be worthless to compute simple statistics that do not take into consideration the unbalanced data we deal with. These metrics are more effective to grasp behavior of the model in such extreme unbalanced situations.
The metric we select as most significant was the Matthews correlation coefficient (MCC). It returns a value between -1 and 1, where 0 indicates a model no better than random guess and 1 is a perfect one.
Our results are presented in the following table.

Sequences Matching			
Model	**Balanced Accuracy**	**F1 Score**	**MCC**
PSSM	0.743	0.483	0.483
HMM 1	0.544	0.119	0.128
HMM 2	0.679	0.527	0.598

The PSSM has an higher balanced accuracy, however the second version of the HMM has a better scores of F1 and MCC.

```
> script model/evaluate_sequences.py
```

9. **Evaluate matching ability for the residues**

Other than matching just the sequences, we evaluated the matching ability at residue level. In order to do so, we took the full sequence of our model and our the positive (annotated) proteins of our ground truth, and built an array composed only of **True** of **False** if the residue fell inside the domain position defined. Then comparing the two parallel arrays we could identify true positives residue matches. This was done only in overlapping cases.
When dealing with false positives we just counted the whole length of the sequence as false positive. The same thing applies for the false negatives.
The number of true negatives residues was, instead, estimated. We calculated them by difference: starting from the total number of residues in *SwissProt* and removing the ones counted previously as true positives, false positives and false negatives.
The guess is in the total number of residues: if SwissProt has 564277 proteins, we estimated 200 million residues in total (with an average of 367 residues per sequence, as reported here).

After building this heavily unbalanced Confusion Matrix we performed the same statistics as before. The results are reported in the following table.

Residues Matching			
Model	**Balanced Accuracy**	**F1 Score**	**MCC**
PSSM	0.567	0.073	0.081
HMM 1	0.508	0.023	0.025
HMM 2	0.521	0.064	0.074

As we can see from the results, the PSSM performs better in terms of matching positions of the residues than the HMM models.

```
> script model/evaluate_residues.py
```

10. Tests

Initially we produced a PSSM and an HMM from the MSA without removing any noise and then we build another HMM model on a cleaned version of the MSA (we removed the final part of the alignment that did not report significant conserved positions).

11. Model selection

Comparing the models we choose the best as the second version of the HMM model, because of matching ability in sequences, even if PSSM was performing slightly better for the residues matching.

Dataset definition

Our best selected model is the HMM model (version 2) generated from de-noised MSA.

1. **Family Structures:** we identified PDB chains from our model sequences using SIFTS data. We just applied the selected filter of at least 80% of coverage.
   We found 10 pdb chains.
2. **Family Sequences:** We built this dataset with the result of the best model in the previous part: extracting the ids from a search against UniRef90.
   We found 53 proteins.

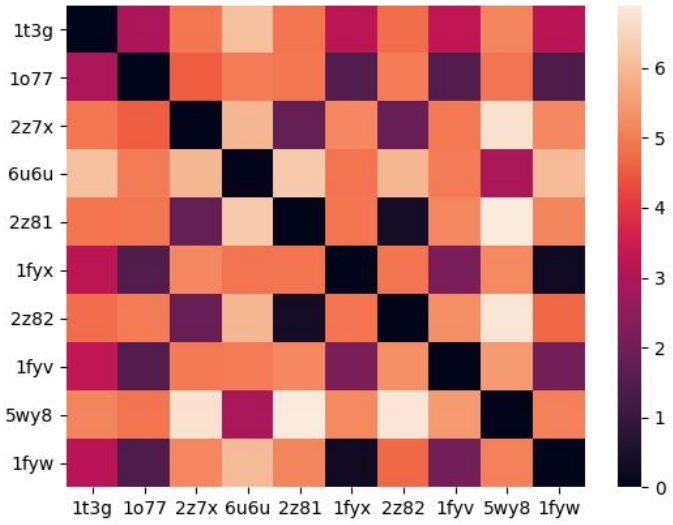
```
> script datasets/create.py
```

Structural characterization

First we downloaded all the pdb files for our family structures.

```
> script structure/download.py
```

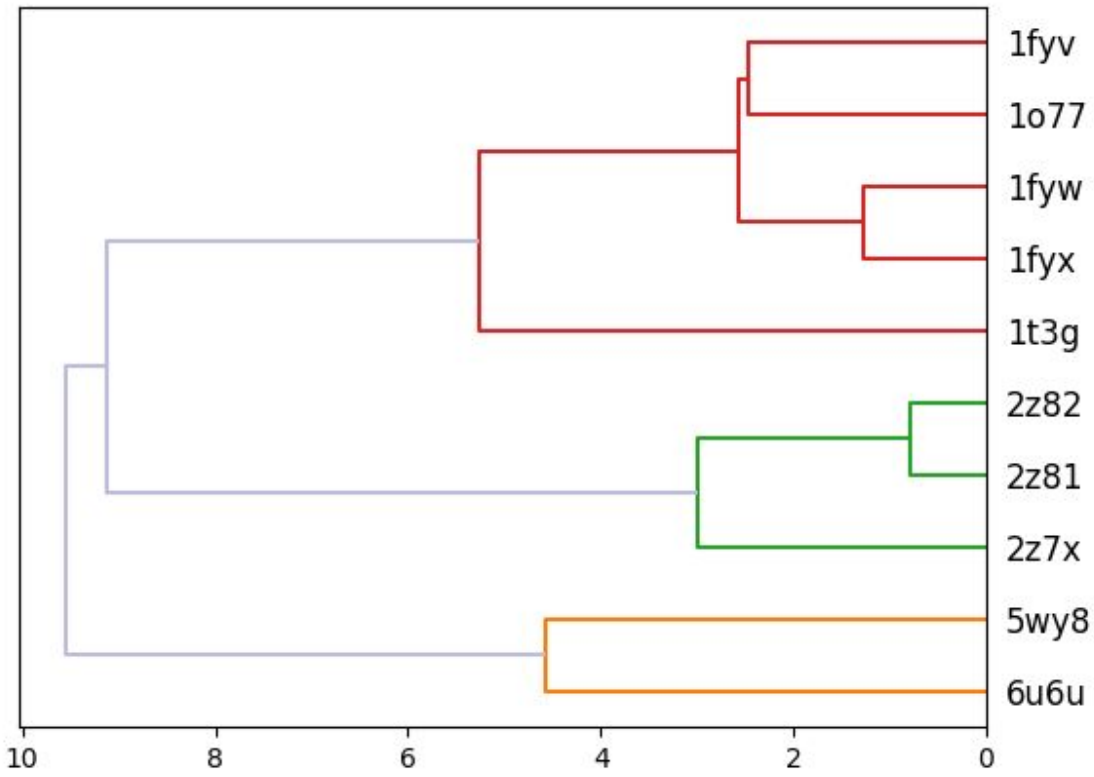
1. **All-vs-All pairwise structural alignment**
   We just performed the all-vs-all alignment with TM-Align software from command line.
2. **Pairwise RMSD matrix & TM-score matrix**
   Then we calculated the RMSD and TM-score matrix from the alignment.



This is the RMSD matrix represented as heatmap.

```
> script structure/align.py
```

3. **Dendrogram**
   To build a dendrogram we use the python libraries *dendrogram* and *linkage* from *scipy.cluster.hierarchy.*



This is the dendrogram of the RMSD matrix. We can see a clear distinction between 3 paths (red, green and orange).

```
> script structure/plot.py
```

4. **Multiple Structural Alignment**
   We performed the multiple structural alignment using the [mTM-Align web interface] where we inserted PDB ids and chains and, when the computation eventually finished, downloaded the alignment in *fasta* format.

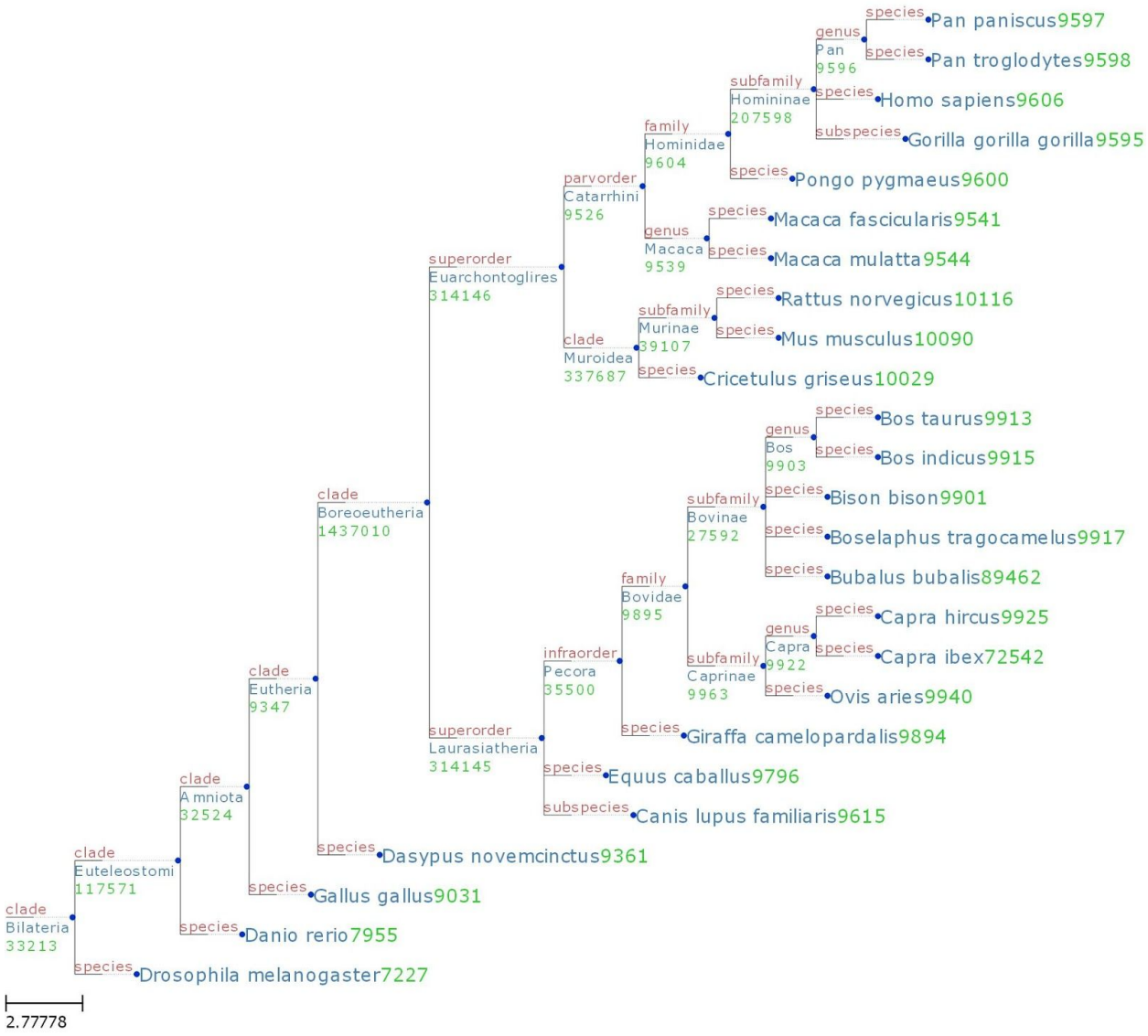
5. **Superfamily/family identification**
   From our PDB sequences we identified 2 CATH superfamilies:
   - **3.40.50.10140**
     Toll/interleukin-1 receptor homology (TIR) domain
     PDB ids:
     - 1fyv
     - 1fyx
     - 1fyw
     - 1o77
     - 1t3g
   - **3.80.10.10**
     Ribonuclease Inhibitor
     PDB ids:
     - 2z82
     - 2z7x
     - 2z81

Taxonomy

1. **Taxonomic lineage**
   Here we parse the dataset used to retrieve the Family Sequences. For each entry (protein) we search in the family_sequences.xml (Uniprot file) and we extract the required data.

```
> script taxonomy/collect.py
```

2. **Taxonomic Tree**
   In this case we use all the taxonomic IDs to create the tree. To do that we used the NCBI taxonomy database. This was a very simple operation due to the fact that we use a specific library (ete3) that performs the search and plots the tree given taxonomy IDs.




```
> script taxonomy/plot.py
```

Functional characterization

1. **Collect GO annotations**
   We collected GO annotations from the UniProt XML file of the family sequences.

```
> script function/collect.py
```

2. **Enrichment**

   In order to parse the gene ontology we used the scripts provided in class along with the *goa_human.gaf* file.
   We computed the *fold increase* and checked the values obtained looking at the *p-values* computed with the *Fisher's exact test*.
   We found 684 enriched terms and they are saved at *data/function/enriched_terms.txt.*

```
> script function/enriched.py
```

3. **Word cloud**

   We used the python library *wordcloud* to create a word cloud starting from a list of enriched terms. Here we report the word cloud generated.


```
> script function/plot.py
```

4. **Most significantly enriched branches**
   In order to get the most significantly enriched branches we consider the hierarchical structure of the GO ontology and first extracted the high level terms of the entire gene ontology. Precisely we selected only the terms at max depth of 1 and 2. Afterwards we filtered out only our enriched terms from the two lists.
   We found 12 terms at depth 2 and 46 terms at the depth 1. Here are the complete lists.

```
# Enriched branches (high level terms) at depth 1
['GO:0060089', 'GO:0002376', 'GO:0050896', 'GO:0065007',
'GO:0044419', 'GO:0023052', 'GO:0003824', 'GO:0022610',
'GO:0032991', 'GO:0009987', 'GO:0005488', 'GO:0110165']
```

```
# Enriched branches (high level terms) at depth 2
['GO:0006955', 'GO:0038023', 'GO:0007165', 'GO:0033218',
'GO:0001816', 'GO:0001775', 'GO:0044297', 'GO:0043235',
'GO:0008289', 'GO:0009607', 'GO:0051707', 'GO:0045321',
'GO:0051606', 'GO:0016787', 'GO:0009605', 'GO:0098796',
'GO:0050789', 'GO:0009986', 'GO:0042221', 'GO:0016020',
'GO:0140352', 'GO:0002252', 'GO:0009719', 'GO:0031224',
'GO:0006950', 'GO:0051716', 'GO:0098552', 'GO:0042995',
'GO:0008219', 'GO:0044281', 'GO:0007267', 'GO:0044877',
'GO:0007154', 'GO:0016192', 'GO:0005576', 'GO:0065009',
'GO:0097367', 'GO:0009628', 'GO:0030054', 'GO:0005515',
'GO:0007155', 'GO:0008283']
```

```
> script function/enriched.py
```

Resources used

https://www.ebi.ac.uk/Tools/msa/tcoffee
https://www.ebi.ac.uk/Tools/hmmer/search/hmmsearch
https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE=Proteins&PROGRAM=blastp&RUN_PSIBLAST=on
http://skylign.org/
https://en.wikipedia.org/wiki/Matthews_correlation_coefficient
http://ftp.ebi.ac.uk/pub/databases/msd/sifts/flatfiles/csv/pdb_chain_uniprot.csv.gz