USER'S GUIDE VERSION 1.0

Valerie De Anda Torres (valdeanda@ciencias.unam.mx)
Augusto Cesar Poot-Hernandez (augusto.poot@iimas.unam.mx)
Bruno Contreras Moreira (bcontreras@eead.csic.es)

Contents

Chapter 1

Software installation

1.1 Obtaining the pipeline

The MEBS software is available as an open-source package distributed from a GitHub repository. Thus, the natural way of installing it is by cloning the repository via the following commands:

```
$ git clone https://github.com/eead-csic-compbio/metagenome_Pfam_score
```

Alternatively, a ZIP file can be downloaded and then unpacked:

```
$ unzip metagenome_Pfam_score-master.zip
```

1.2 Software dependencies

There are a few external packages which are required in order to run this pipeline. They have been tested in Linux (Ubuntu) environments.

1. Perl 5, preinstalled in most Linux systems
2. InterproScan
3. HMMER v3

In addition, the Python 3 interpreter and a few modules are required as well:

1. Matplotlib v1.4 or greater
2. Numpy
3. Pandas

4. Scikit-learn
5. mpltoolkits
6. Jupyter-notebook

Under ubuntu systems you can use the following comands (Tested in Ubuntu 16.04.2)

```
$ sudo apt-get install python3 python3-pip python3-matplotlib \
 ipython3-notebook python3-mpltoolkits.basemap
$ sudo pip3 install -U pip
$ sudo -H pip3 install --upgrade pandas
$ sudo -H pip3 install --upgrade numpy
$ sudo -H pip3 install --upgrade scipy
$ sudo -H pip3 install --upgrade seaborn
$ sudo -H pip3 install -U scikit-learn
```

Chapter 2

Running the pipeline

The pipeline comprises 4 steps, which are described in Figure 2.1 taking the Sulfur cycle as an example:

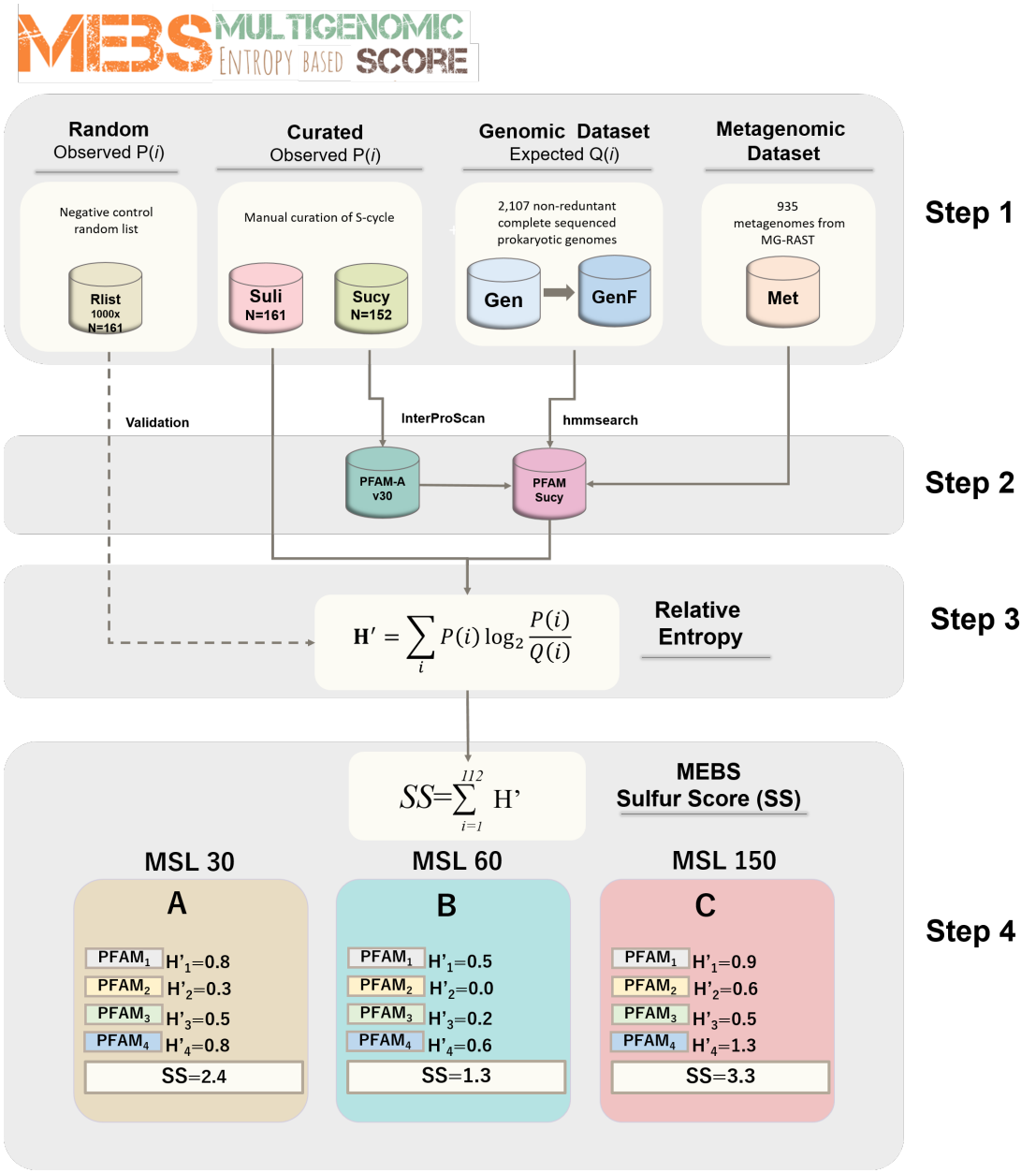

Figure 2.1: The MEBS pipeline comprises four steps. Steps 1-3 (**"advanced mode"**) are required to train an entropy-based classifier, and require expert intervention and data curation (**step 1**), annotation of Pfam protein domains (**step 2**) and computing relative entropies of domains annotated in proteins related to the cycle of interest (**step 3**). Once these steps are completed, the trained model can be used to score arbitrary genomes or metagenomes (**step 4**, which is called **"simple mode"**). In the example, the Sulfur Score (SS) is computed for three metegenomic peptide sets with mean fragment sizes of 30, 60 and 150, respectively.

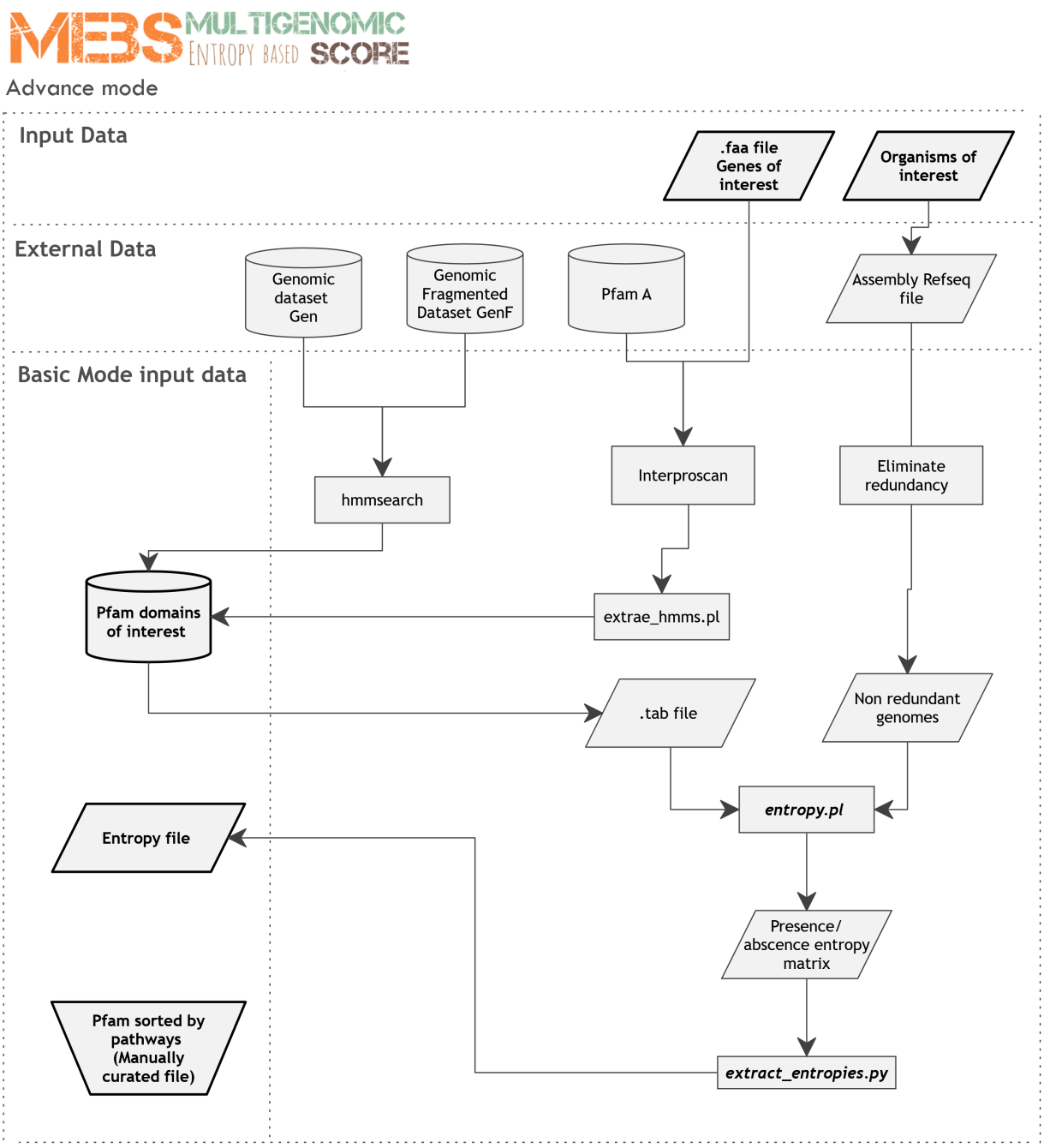

Figure 2.2: The MEBS pipeline (**"advanced mode"**)in flowchart diagram

2.1 SIMPLE MODE

Arbitrary FASTA peptide files can be scored by the user. These can be extracted from annotated genomes or metagenomes derived from Microbial Gene Prediction sofware (i.e `Prodigal`). Alternatively,

sequences from `RefSeq` or `MG-RAST` databases can be used as well.

Sample genomes and metagenomes files are provided in folders: `test_genomes/` and `test_metagenomes/`, respectively. They can be scored in terms of their **Sulfur cycle** metabolic machinery with BASH scripts *score_genomes.sh* and *score_metagenomes.sh*, as summarized in Figure 2.3. These scripts require `hmmsearch` to be installed.

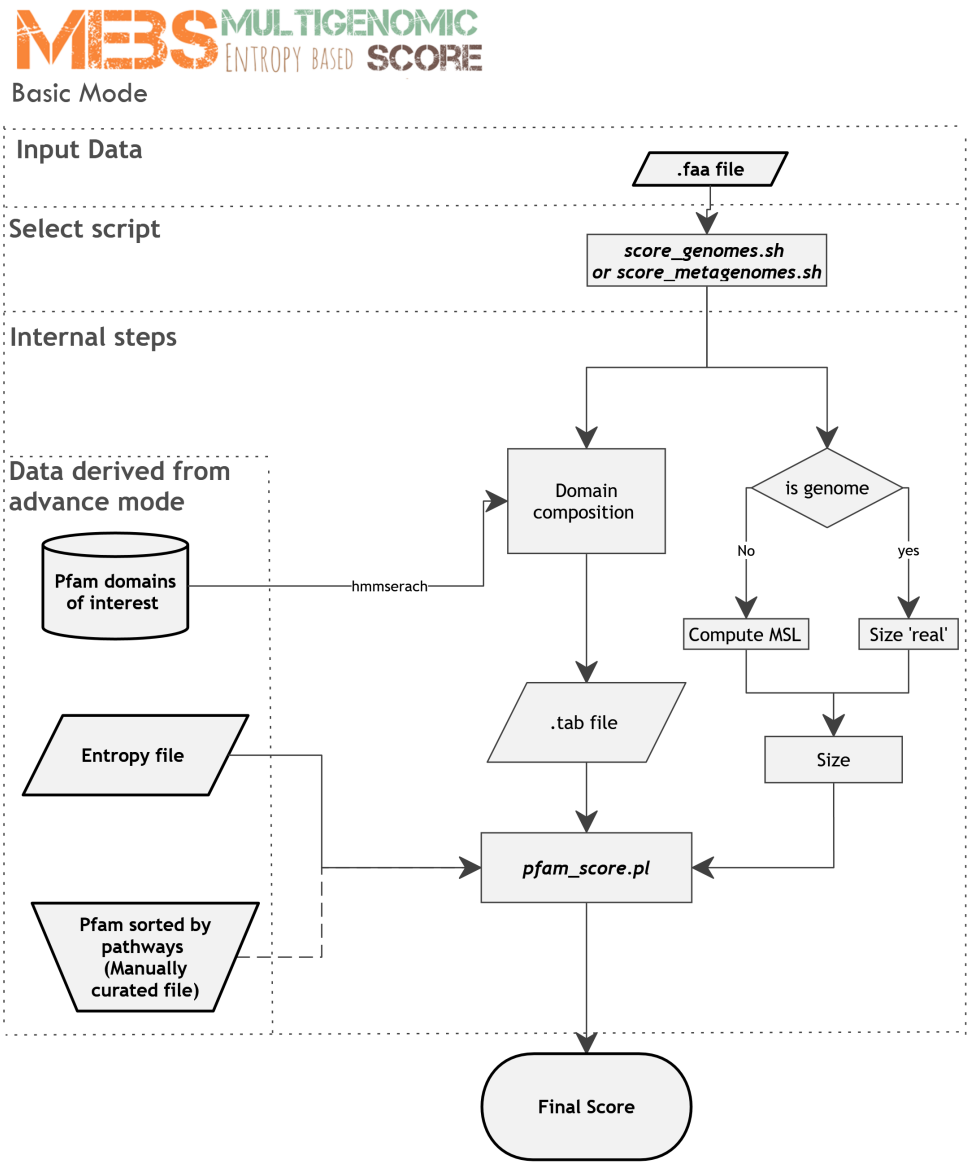

Figure 2.3: Running MEBS simple mode

More generally, these scripts can be used to analyze a directory containing peptide FASTA files of encoded proteins/fragments with .faa extension. Examples of use would be:

```
$ ./score_genomes.sh test_genomes
or
$ ./score_metagenomes.sh test_metagenomes
```

2.2 ADVANCED MODE

2.2.1 Step 1: Databases and manual curation

2.2.1.1 Curated input

The following files must be curated by the user in order to train his/her own data.

1. MultiFASTA file with peptide sequences related to the metabolism of interest. See for instance the contents of the **Sucy** FASTA file, provided with this software

   (`sulfur_data_test/input_sulfur_data/sucy_database_uniprot.fasta`):

   ```
   >sp|Q54506|SAT_RIFPS Sulfate adenylyltransferase OS=Riftia...
   MIKPVGSDELKPLFVYDPEEHHKLSHEAESLPSVVISSQGPRVSSMMGAGYFSPAGFMNV
   ADAMGAAEKMTLSDGSSSCSVLCLLENTDAIGDAKRIALRDPNVEGNPVLAVMDIEAIEE
   VSDEQMAVMTDKVYRTTDMDHIGVKTFNSQGRVAVSGPIQVLNFSYFQADFPDTFRTAVE
   IRNEIKEHGWSKVVAFQTRNPMHRAHEELCRMPMESLDADGVVVHMLLGKLKKGDIPAPV
   RDAAIRTMAEVYFPPNTVMVTGYGFDMLYAGPREAVLHAYFRQNMGATHFIIGREPPAWV
   TTTVPSTPRPSSMTKCQRAPWRSRSSCRPHGLLQEAEQDCDDARRAGSHQGRLRTALRHQ
   GREMLGQGIAPPPEFSRPEVAKILMDLLPVHQQLILIWFSGKTRPGVGRWRVFLCAAGAL
   WPEAVAVANMEKRSSTG
   >tr|O33997|O33997_ALLVI Adenylylsulfate reductase alpha subunit...
   MAYKTIIEDGIDVLVVGAGLGGTGAAFEARYWGQDKKIVIAEKANIDRSGAVAQGLYAIN
   ...
   ```

2. List of genomes with know/documented metabolic capabilities. See the example **Suli** file in `sulfur_data_test/input_sulfur_data/sulfur_list_genomes.txt`:

   ```
   GCF_000006985.1 Chlorobium tepidum TLS
   GCF_000007005.1 Sulfolobus solfataricus P2
   GCF_000007305.1 Pyrococcus furiosus DSM 3638
   GCF_000008545.1 Thermotoga maritima MSB8
   GCF_000008625.1 Aquifex aeolicus VF5
   GCF_000008665.1 Archaeoglobus fulgidus DSM 4304
   ...
   ```

2.2.1.2 Genomic dataset

In order to benchmark the classifier, we use `RefSeq` database to obtain well-annotated, reference genomic prokaryotic sequences. Then, we reduce redundancy using the web-based tool described in Moreno-Hagelsieb et al. 2014.

1. If the user wants to use the same dataset to train his/her own data, it can be downloaded from GitHub. In this case the following steps can be skipped.

2. On the contrary, if the user wants to repeat all the steps with the up-to-date contents of Refseq, the following steps are needed. Before you start make sure you have at least of 110Gb of free space in your drive to store the genomic and genomic fragmented datasets.

2.2.1.3 Building your own genomic dataset (Gen)

Visit the Genome Clusters server and set the following "relaxed" parameters:

```
GSS / DNA Signature = GSSb
GSS threshold = 0.95
DNA-signature threshold = 0.01
Sort by size or overannotation = largest
Results style = simple list
```

In our Sulfur cycle benchmark, the RefSeq release of August 2016 contained 4085 prokariotic genomes, which were reduced to 2,114 clusters and eventually 2,107 non-redundant genomes. Our nr list can be inspected at `Gen/GSSb-0.95.txt`:

```
Group 1 GCF_000005825
Group 2 GCF_000005845,GCF_000006925.....
Group 3 GCF_000006175
Group 4 GCF_000006605
Group 5 GCF_000006625,GCF_000019345,GCF_000828735
...
```

Follow these steps in order to get a multiFASTA file of non-redundant set of genomes.

1. Parse the former list in order to obtain the identifiers of non-redundant genomes

```
$ cd Gen
$ sed 's/,/\t/g' GSSb-0.95.txt  | sed 's/ /\t/g'  | cut -f 3  >
list_nr_genomes_21122016.txt
```

2. Obtain the assembly data from the selected non redundant genomes

```
$ wget "ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq
/assembly_summary_refseq.txt"
$ for i in `cat list_nr_genomes_21122016.tx` ; do grep $i
assembly_summary_refseq.txt >> assembly_refseq.nr2016.txt ; done
```

3. Get the download links

```
$ less assembly_refseq.nr2016.txt | cut -f 20 \
| sed 's/$/\/*.faa.gz/g' > assembly_refseq.nr2016.download.txt
```

4. Download the genomes in `faa` format

```
$ wget -i assembly_refseq.nr2016.download.txt
$ gunzip *.gz
```

5. Before generate a multiFASTA file, it is important to change the headers of all the genomes for latter purposes, described in section 2.2.2

```
$ for i in *.faa ; do perl -lne 'if(/^>(\S+)/){ print ">$1 [$ARGV]"}
else{ print }' $i > $i.named.faa; done
```

6. The previous command will change the headers from

```
>WP_003320558.1 MULTISPECIES: ATP-binding protein
[Bacillus]
MNEQIQAYAKRLKLSWIRENFNQIEAETNEEYLLKLFEKEVQNREERKVNLLLSQ
AQLPKTGSTPFQWEHIQIPQGIERT
```

to this

```
>WP_003320558.1 [GCF_000005825.2_ASM582v2_protein.faa]
MNEQIQAYAKRLKLSWIRENFNQIEAETNEEYLLKLFEKEVQNREERKVNLLLSQ
AQLPKTGSTPFQWEHIQIPQGIERT
```

7. Generate the multiFASTA file

```
$ cat *.named.faa > genomes_refseq_nr_22122016.faa
```

8. Generate one-line multiFASTA file using the following perl snippet

```
$ perl -lne 'if(/^(>.*)/){ $head=$1 } else { $fa{$head} .= $_ } END{
foreach $s (sort(keys(%fa))){ print "$s\n$fa{$s}\n" }}'
genomes_refseq_nr_22122016.faa > genomes_refseq_nr_22122016.1.faa &
```

2.2.1.4 Building your own Genomic fragmented dataset (GenF)

Using the multiFASTA file containing the non-redundant genomes, then the genomic fragmented set can be generated using script *get_protein_fragments.pl*.

Depending on the Mean Size Length (MSL) observed in the fragmented metagenomic dataset, the the Genomic fragmented dataset is generated. We use seven categories to fragment the Genomic dataset: 30, 60, 100, 150, 200, 250 and 300. These sizes were choosen due to the observed bins in the Metagenomic dataset, see details in section 5. In order to see the help page you can type:

```
$ perl scripts/get_protein_fragments.pl

Program to produce random fragments of proteins in input file
with size and coverage set by user.
usage: get_protein_fragments.pl [options]
 -help    brief help message
  -inFASTA    input file with protein sequences in FASTA format
 -outFASTA    output file with protein fragments in FASTA format
  -size    desired size for produced random fragments
  (integer, default 100)
  -cover    desired protein coverage of produced fragment
           (integer, default 1)
```

We can now loop along the desired fragment sizes:

```
$ for i in 30 60 100 150 200 250
perl get_protein_fragments.pl -inFASTA genomes_refseq_nr_21122016.1.faa \
-oUTFASTA genomes_refseq_nr_21122016_size${i}_cover10.faa -size $i \
-cover 10 ; done
```

Then create a new directory containing the *in silico* sheared FASTA files.

```
$ mkdir sulfur_data_test/GenF && mv *cover* sulfur_data_test/GenF
```

Using the above commands, the following files should be generated (Table 2.1).

Table 2.1: Description of the genomic and genomic fragmented datasets

Details	File name	Size File
nr-genomes	genomes_refseq_nr_22122016.1.faa	2.7G
nr-genomes size 30	genomes_refseq_nr_21122016_size30_cover10.faa	7.8 G
nr-genomes size 60	genomes_refseq_nr_21122016_size60_cover10.faa	9.8 G
nr-genomes size 100	genomes_refseq_nr_21122016_size100_cover10.faa	13 G
nr-genomes size 150	genomes_refseq_nr_21122016_size150_cover10.faa	16 G
nr-genomes size 200	genomes_refseq_nr_21122016_size200_cover10.faa	18 G
nr-genomes size 250	genomes_refseq_nr_21122016_size200_cover10.faa	20 G
nr-genomessize 300	genomes_refseq_nr_21122016_size200_cover10.faa	22 G

2.2.1.5 Metagenomic dataset

If you want to train your classifier using public available metagenomes, you can use the following commands to download them from `MG-RAST` server version 3.6. Otherwise, if you have your own data, skip this step, and compute the Mean Size Length of your metagenomes (see section 5).

2.2.1.6 Building your own metagenomic dataset

Generate a list of metagenomes of interest. We selected only those metagenomes that meet the following conditions:

1. Public and available metagenomes
2. Available metadata associated
3. Environmental samples (isolated from defined environments o features: rivers, soil, biofilms, etc), discarding the microbiome associated metagenomes (i.e human, cow, chicken)
4. We also included 35 private metagenomes unpublished derived from sediment, water and microbial mats from Cuatro Cienegas Coahuila (De Anda, in progress).

Using the above mentioned conditions, a total of 935 id-numbers were saved in list format
See the example file `sulfur_data_test/Met/id_metagenomes.txt`:

```
4524971.3
4451035.3
4557986.3
4441021.3
...
```

2.2.1.7 Considerations

1. If you decide to use the data in the *id_metagenomes.txt* file, please be aware that this process will take several hours, and at least 500 Gb of drive space.
2. Otherwise you can change the id-numbers from those of your interest.
3. Make sure you are inside folder `sulfur_data_test/Met/`
4. Get the corresponding encoded protein sequences using the MG-RAST API.

```
$ for line in  `cat id_metagenomes.txt` ; do wget
"http://api.metagenomics.anl.gov/1/download/mgm$line?file=350.1" \
-O $line ; done

#Check the files

$ head 4524971.3
>FA4S SCL02FZQ74_1_237_-
LGRPIDIESLDVSFWGGLGVVLGNVVISNPEDMPGDTLMVAKEI
DVKLQLWPLLSSEVRADRFIINDPTI
>FA4SSCL02ILJM5_1_200_-
YIYLTLVEFKGQAAGLRERESIQHFQWSDKQVSKKEGSFGVDHI
WYLQGIFMCSSIQKDYPKIK

#Make sure that all the files were downloaded correctly

$find -empty | wc
```

5. Make sure you are located in the same directory containing the peptides from the downloaded metagenomes and compute the Mean Size Length (`MSL`) of each metagenome using the following one-line perl script:

```
$ for FILE in *; do perl -lne 'if(/^(>.*)/){$h=$1}else{$fa{$h}.=$_}
END{ foreach $h (keys(%fa)){$m+=length($fa{$h})};
printf("%1.0f\t",$m/scalar(keys(%fa))) }' $FILE; echo $FILE; done >
MSL.tab
```

6. See the example of the MSL output from all the 935 metagenomes

```
$ head  sulfur_data_test/Met/MSL.tab

52 4524971.3
79 4451035.3
75 4441022.3
```

7. Plot the MSL sizes 2.4

```
Open R from the terminal
$ R
> data<-read.table("sulfur_data_test/Met/MSL.tab", header=F, sep="\t")
> pdf("hist.msl.pdf")
> hist(data$V1,main= "MSL metagenomic dataset",xlab= "MSL in aa")
> dev.off()
> quit()
$ evince hist.msl.pdf  &
```

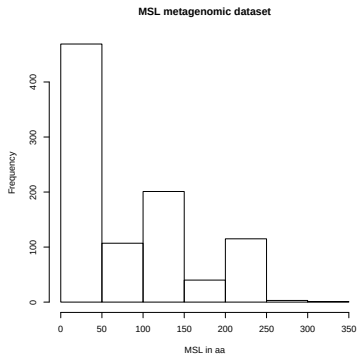

Figure 2.4: MSL histogram of the metagenomic dataset

2.2.1.8 Building your own random dataset

The random dataset consist of arbitrary number of lists containing random genomes not particularly enriched in the metabolism of interest. For this purposes, consider the same number of genomes from your input list. First you need to know how many genomes are you given as input, as described in section 2 and then shuffle the assembly file from `RefSeq` to get random accession numbers.

```
#1. First create a new directory that will contain all the lists
$ mkdir random samples
```

```
#2. Count the number of genomes in your input list
$ num=‘less sulfur_data_test/input_sulfur_data/  sulfur_list_genomes.txt | \
wc -l‘

#3. Create 1000 list of random accession numbers
$ for ((i=1;i<1001;i+=1)); \
do cat Gen/assembly_refseq.nr2016.txt \
| cut -f 1,8 | shuf -n $num  > random_samples/random$i.txt  ; done
```

The random created lists, will be used in section 2f

2.2.2 STEP 2: Domain annotation

1. Annotate the domain composition of your input proteins (See example in section 2.2.1.1) using `InterproScan` and the current release of `PfamA` database (updated December 2016)

   ```
   #Run interproscan

   $ interproscan.sh -appl PfamA-30.0 \
   -i  sulfur_data_test/input_sulfur_data/sucy_database_uniprot.fasta
   \ -f tsv -pa  -iprlookupD
   ```

2. Check the output example, using following the output information of InterproScan

   ```
   $ less -S sulfur_data_test/sucy_database_uniprot.fasta.tsv
   P38038  fb5aec671520cf6aab02a5a3862c2aeb 599  Pfam  PF00258 ..
   P38038  fb5aec671520cf6aab02a5a3862c2aeb 599  Pfam  PF00175 ..
   ```

3. In order to get the family domains of the input proteins, make sure that you have the current release of Pfam database. Usually is installed with Interproscan. You also can get the current Pfam release using the following command line.

   ```
   $ wget \
   ftp://ftp.ebi.ac.uk/pub/databases/Pfam/current_release/Pfam-A.hmm.gz
   gunzip Pfam-A.hmm.gz
   ```

4. Get the specific `Pfam` domain accession numbers of the input proteins. Use the output file from `InterproScan`

```
$ cut -f 4,5 \
sulfur_data_test/sucy_database_uniprot.fasta.tsv >id_interpro.txt
```

5. Obtain the specific markov models of the input proteins.Run the script *extract_hmms.pl*. The latter use as input the id_interpro.txt and the PfamA.hmm. Make sure you have those files in the same directory. And the names are exactly the same, otherwise the script will not work.

```
$ perl scripts/extract_hmms.pl

# Pfam
# hmms = 112 #pfam version 30.
```

This will generate an output file named 'my_Pfam.hmm. We have renamed the file, which is provided precomputed at sulfur_data_test/my_Pfam.sulfur.hmm

Note that Superfamily and TIGRFAM HMMs can also be used, but are commented out in the script since in our tests they produced huge outfiles (SF) or largely redundant with Pfam (TIGR).

6. Annotate Pfam domains in the "omic"-data sets using HMMER.

Pfam annotation in the multifasta file of non-redundant genomes

If you build your own non redundant dataset, change the <*.faa> input file by your file name. Otherwise, you could use file containing 2107 genomes as described above in section 2.2.1.2

```
$ hmmsearch --cut_ga -o /dev/null  --tblout
genomes_refseq_nr_22122016.fa.pf.tab
my_Pfam.hmm genomes_refseq_nr_22122016.1.faa  &
```

Optionally, you can compute hmmsearch for each genome separately, in order to calculate their Score as explained in detail in 2.2.5

Pfam annotation in the metagenomic dataset

```
$ for i in sulfur_data_test/Met/* ; do \
hmmsearch --cut_ga -o /dev/null --tblout \
$i.out.hmmsearch.tab my_Pfam.hmm  $i; done &
```

2.2.3 STEP 3: Relative Entropy

We used a derivative of the Kullback-Leibler divergence, also known as relative entropy (H') to measure the difference between two probability distributions P(i) and Q(i). See 2.1

$$H' \ P(i) log_2 \frac{P(i)}{Q(i)} \tag{2.1}$$

In this context, P(i) represents the total number of occurrences of protein family i in sulfur-related genomes (observed frequency), while Q(i) represents the total number of occurrences of that family in the genomic dataset (expected frequency). The relative entropy H', in bits, captures the extent to which a family informs specifically about sulfur metabolism. H' values that are close to 1 correspond to the most informative families (enriched among sulfur-related genomes), whereas low H' values (close to zero) describe non-informative families. Negative values correspond to protein families observed less than expected

Requirements to compute relative entropies

1. A hmmsearch TSV outfile with the results of scanning a collection of Pfam domais against a large set of(non- redundant) genomes (output from hmmsearch in the omic datasets) described in section 6. We provided the outfile derived from the results of scanning the 112 Sulfur related models against the 2,107 non redundant genomes described in section 6 Download the file from the GitHub-release

2. A list of selected accessions of genomes interest to compute entropy see example file `sulfur_data_test/input_sulfur_data/sulfur_list_genomes.txt`, described in section 2

3. An optional list of `RefSeq` assembly annotations to print scientific names instead of accession codes. See example file `Gen/assembly_refseq.nr2016.txt` described in section 2

Computing the relative entropy in the genomic dataset

The entropies matrices precomputed for the genomic (Gen) and genomic fragmented (GenF) are provided in the `sulfur_data_test/entropies_matrix` directory. We provided the data to compute the matrices for the Gen dataset.For space reasons GenF is not provide, but it can be easily generated as explained in previous sections 2.2.1.4. The entropies matrices are obtained with the script *entropy.pl*. Firs Run the script help:

```
$ perl scripts/entropy.pl : usage: scripts/entropy.pl
<pfam_hmmsearch.tab> <accession list (ie Suli)> <RefSeq
list, optional>
```

Create a directory containing the obtained matrices

```
$ mkdir entropies_matrix
```

Obtain the entropy matrix from the Gen dataset (Real size).

```
$ perl scripts/entropy.pl \
genomes_refseq_nr_22122016.1.faa.out.hmmsearch.tab
sulfur_data_test/input_sulfur_data/sulfur_list_genomes.txt
Gen/assembly_refseq.nr2016.txt >
entropies_matrix/genomes_refseq_nr_22122016.fa.pf.tab.csv
```

After computing the relative entropies from your own data, you will get the following files:

1. A matrix of occurrence of Pfam domains across genomes
2. Entropy estimates of each scanned Pfam domain with respect to the selected accessions

Observing the distribution of the relative entropies

In order to plot the results obtained with the *entropy.pl* script, the following scripts are needed:

1. *entropies.py*: The first script requires all the matrix that contains the relative entropies computed in the genomic and genomic fragmented data-set (sizes 30, 60, 100, 150, 200, 250, 300). The matrix are located in entropies_matrix directory. The scripts returns a tabular list of the entropies. Make sure that the names of the matrix in csv format follow this pattern ˂size([0-9]+)_˃ otherwise the script cannot be computed. Besides, all the files need to have the same number of domains (profiles) in the same column order. This script assumes that these considerations are true, so it cannot find errors in the input files format.

   ```
   $ python3 scripts/extract_entropies.py entropies_matrix/
   ```

   The latter script will generate the text file `entropies_matrix_entropies.tab`

   ```
           real    30      60      100     150     200 ...
   PF00005 -0.001  0.001   -0.001  -0.001  -0.001  -0.001 ...
   PF00009 -0.001  -0.014  -0.001  -0.001  -0.001  -0.001 ...
   PF00034 -0.119  -0.195  -0.183  -0.153  -0.115  -0.106 ...
   ```

2. *plot_entropy.py*: In order to observe the results of the latter tabular file, this scripts will generate 5 different plots:

```
Usage:
 $ python3 plot_entropy.py data.tab [random perc5] [random perc95]

Run the script
 $ python3 plot_entropy.py entropies_matrix_entropies.tab
```

After running the script, the following figures are generated:

(a) **entropies_matrix_entropies.tab_bar.png.** Barplot of the distribution of each Pfam relative entropy, in the different genomic fragmented sizes. At the top of the figure are the Pfam's with highest values, and the bottom are the lowest entropies values . (Figure 2.5)

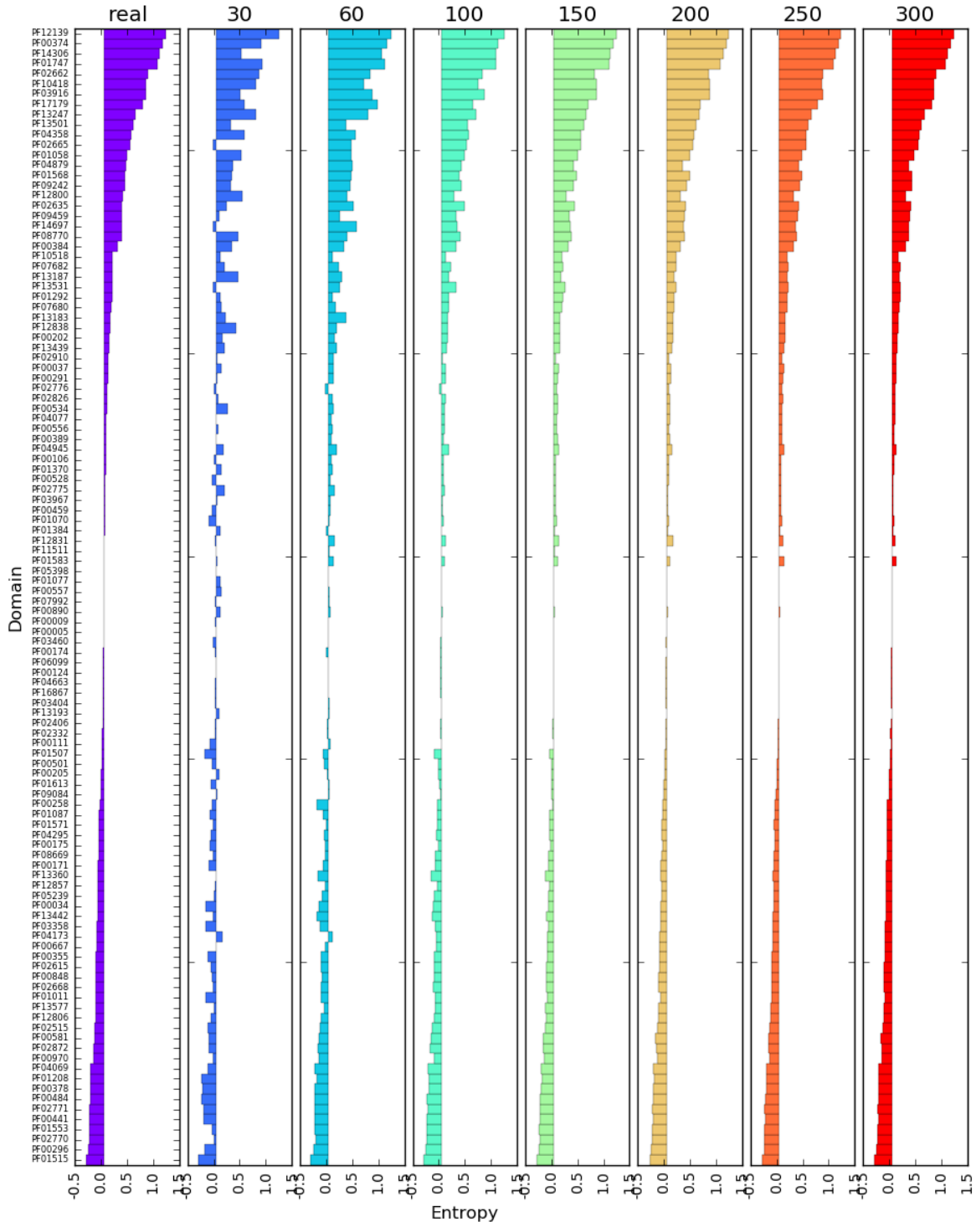

Figure 2.5: Distribution of relative entropies of Pfam domains in Gen and GenF.

(b) **entropies_matrix_entropies.tab_hmap.png**. Heatmap of the distribution of each Pfam relative entropy, in the different genomic fragmented sizes. Red values are the highest

entropies and blue values the lowest. (Figure 2.6)

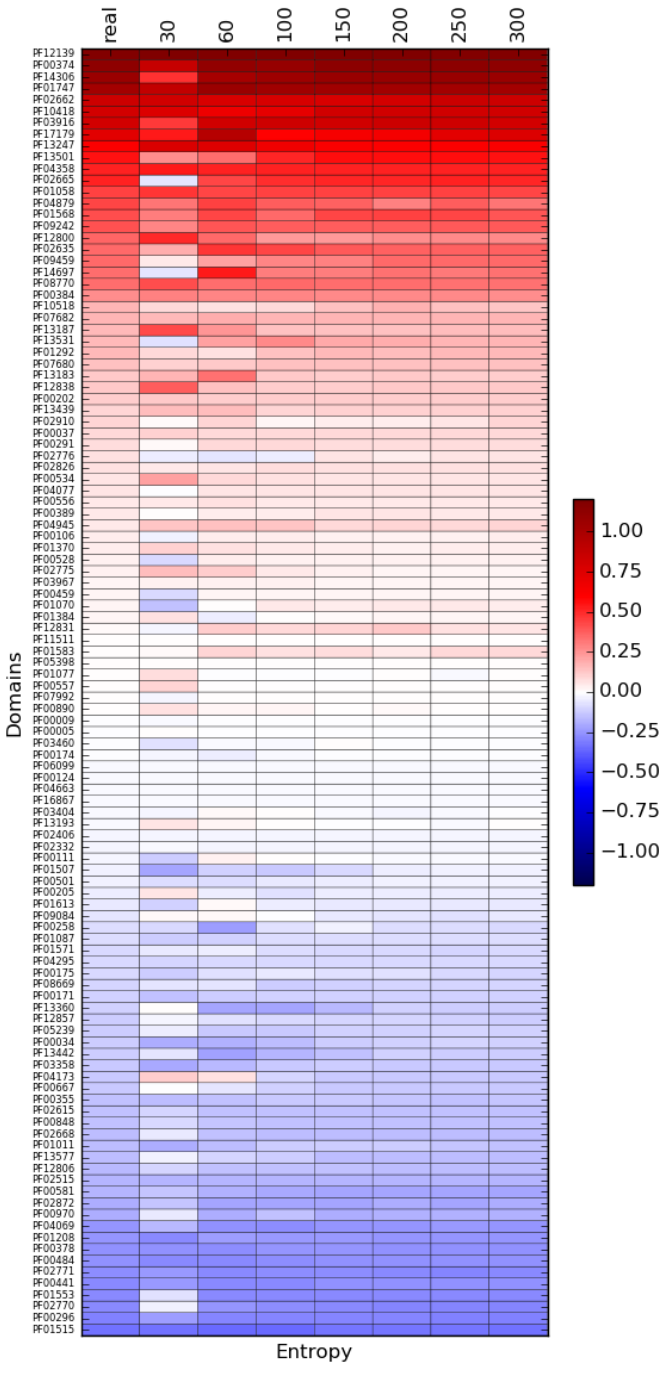

Figure 2.6: Distribution of relative entropies of Pfam domains in Gen and GenF.

(c) **entropies_matrix_entropies.tab_scatter.png**. Scatter plot showing the mean dispersion of each Pfam H' (x axis),versus the standard deviation of the values obtained in the genomic

fragmented dataset (y axis). In this sense the low standard deviation indicates that the H' values are similar across several datasets of variable sizes, which is useful in the metagenomic dataset. In this sense, the H' of this specific Pfam's, are not affected by the size of the metagenome (either read peptides of 30 aa or 300). In the other hand, high standard deviation indicates that H' is affected by the size. In this way, high H', and low standard deviation, points the most informative Pfam's that could be used as molecular marker genes in metagenomes of variable sizes. (Figure 2.7)

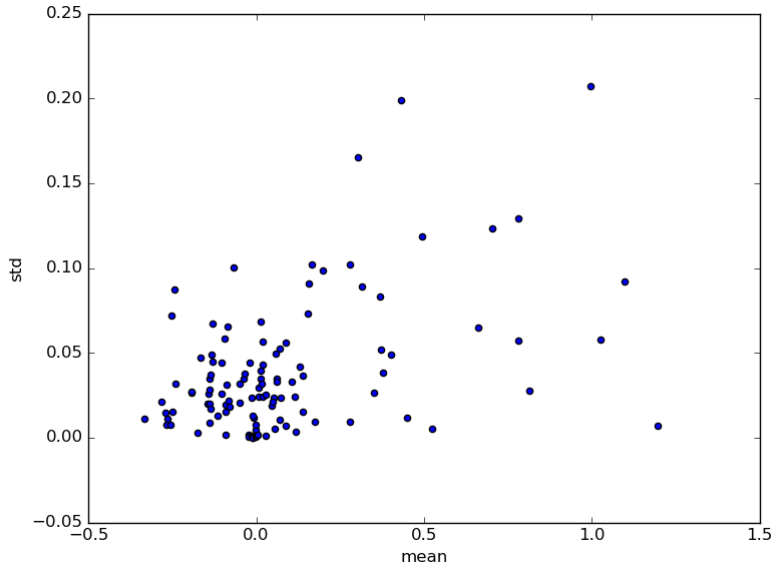

Figure 2.7: Dispersion of Pfam entropies.

(d) **entropies_matrix_entropies.tab_entropy_hist.png**. Histograms of the distribution of the H' in the different genomic fragmented sizes. (Figure 2.8)

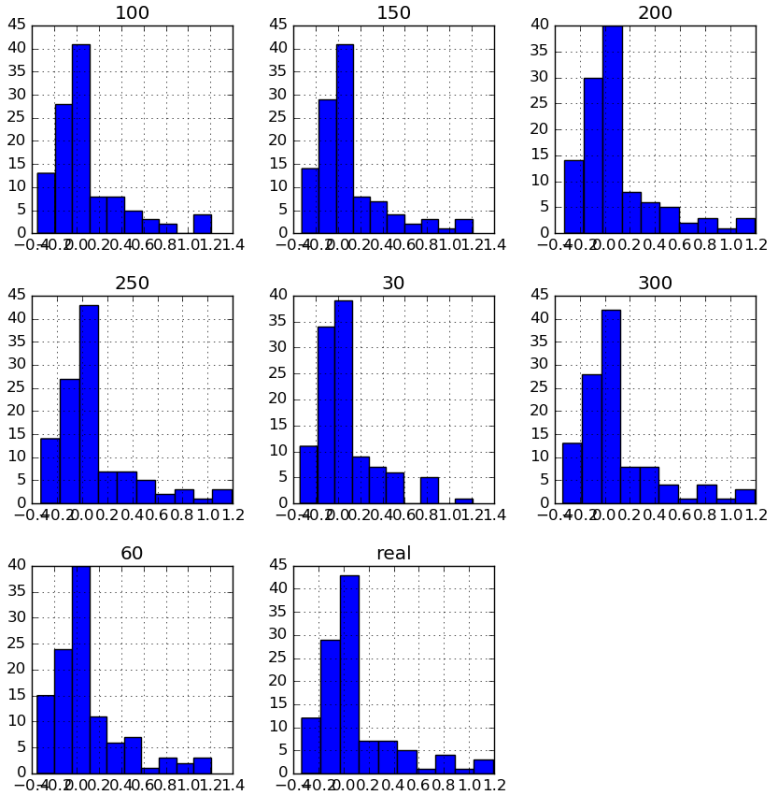

Figure 2.8: Distribution of Pfam entropies in Gen and GenF datasets.

(e) **entropies_matrix_entropies.tab_differential.png**.

Entropy difference of each Pfam H' with respect to real, in order to observe the degree of change from the previous one. The differential was calculate along the MSL

$x_i - x_{i-1}$.

The data was normalized with respect to real values, then the differential was plotted according to each size (MSL). The highest entropy difference with respect to real values is obteined in sizes of 30 and 60, indicating that above this values, the H' of each Pfam are maintained across several datasets of variable sizes (>60). (Figure 2.9)

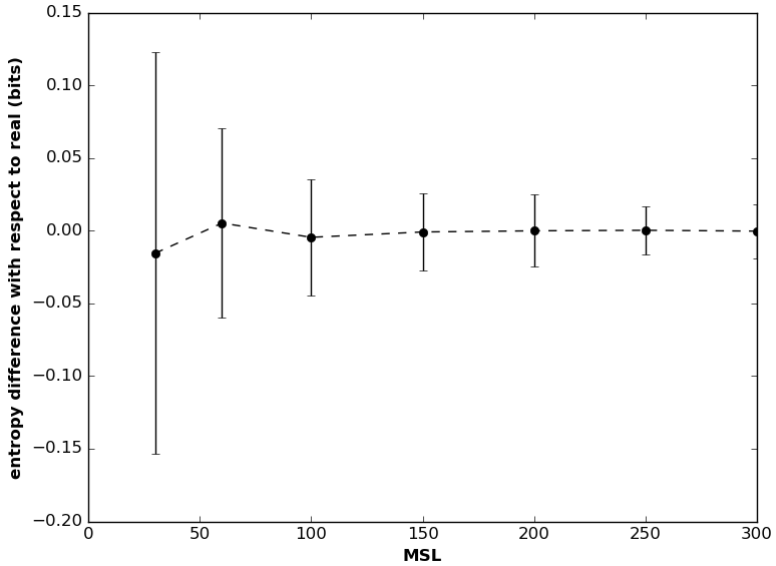

Figure 2.9: Differential plot

(f) **entropies_matrix_entropies.tab_prof_box.png**.Boxplot the distribution of each Pfam relative entropy, in the different genomic fragmented sizes. Middle blue line indicates cero values. Black lines indicates the percentile data (5% and 95%) obtained obtained in the random test. (Figure 2.10). See further details in Random entropies.

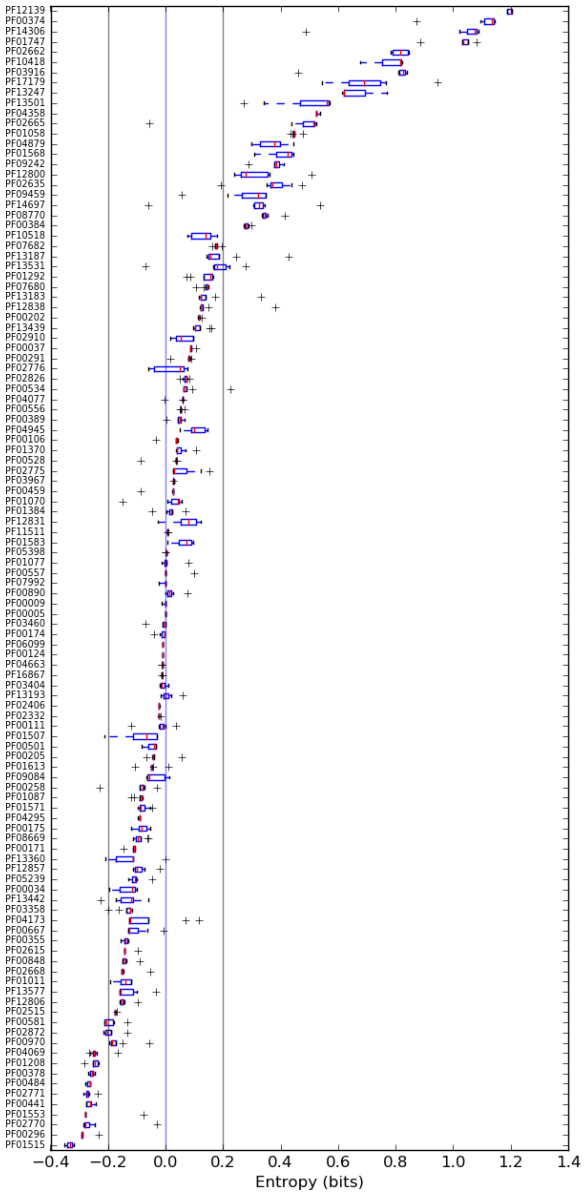

Figure 2.10: Boxplot distribution of Pfam relative entropies in Gen and GenF

2.2.4 STEP 3: Relative entropy of random datasets

Generate the entropies from each of the the random list computed in section 2.2.1.8.

i. First, compute the random entropies from Gen dataset

```
$ for i in random_samples/*.txt; do perl scripts/entropy.pl
genomes_refseq_nr_22122016.1.faa.out.hmmsearch.tab \
$i > $i.csv; done
```

ii. Then use the GenF to compute the random entropies considering fragmented sizes You can either save these command lines into a bash script.

```
#!/bin bash
for msl in 30 60 100 150 200 250 300
do
for file in random_samples/*.txt; do perl scripts/entropy.pl
genomes_refseq_nr_22122016_size"${msl}"_*.tab
$file > $file.$msl.csv;  done ; done
echo "Done, please check the csv matrices in random_samples"
```

iii. Extract the entropies values of all the 8,000 matrices, and get a tabular format file of each Pfam and the corresponding H' value on each test (1..100). Use the script *extract_random_entropies.pl*

```
perl ../scripts/extract_random_entropies.pl -matrixdir
random_samples/

# extract_random_entropies.pl call:
# -matrixdir random_samples

# merged file of MSL=real
(replicates=1000, random_real.real.tab)
```

iv. Generate a new folder and move all the generated files

```
mkdir tab_files_random && mv *.tab tab_files_random
```

v. Plot the tabular files output using the following script

```
python3 scripts/plot_random_entropies.py random_real.real.tab
```

vi. Using the above mentioned script, we obtained both: 1) the percentile data of all the tabular files computed for each size (MSL) (Table 2.2), and 2) the boxplot distribution of each Pfam H' value in the the random See the complete description of the output files.

```
.
1) Percentile distribution at 5% and 95%  of each Pfam H',
in the 1000 random matrices, for example:

PF13501  5-percentile= -0.049  95-percentile= 0.077
PF13531  5-percentile= -0.058  95-percentile= 0.058

2) The total min 5-percentile and max 95-percentile
# min  5-percentile= -0.091
# max 95-percentile= 0.101

3)Boxplot distribution of each Pfam H', indicating the
lines of the max an min percentile distribution.
```

After running the entropy random test using the Sulfur cyle data, you will get the following percentile distribution of the Gen and GenF. (See Table 2.2)

Table 2.2: min 5 and max 95 percentile distribution of Pfam's H' in each MSL

MSL	5 percentile	95 percentile
Real	-0.091	0.101
30	-0.086	0.105
60	-0.09	0.105
100	-0.088	0.1
150	-0.09	0.103
200	-0.089	0.105
250	-0.09	0.106
300	-0.09	0.1

2.2.5 STEP 4: Entropy Score and interpretation

2.2.5.1 Requirements

Due to the variation in peptide size within metagenomic datasets (see Figure 2.4, it would be computationally expensive to fragment the genomic datasets in all the observed sizes. Therefore, we chose to fragment them according to the observed sizes/bins in the histogram (see (Figure 2.4): 30,60,100,150,200,250 and 300. Taking this into account, we propose a range of ± 15 aminoacid residues above or below the size to compute the Score. Accordingly, entropy scores for a metagenome of MSL of 40 would be computed using pre-computed entropies of the genomic fragmented (GenF) dataset of size 30. See Table 2.3

Table 2.3: MSL selection of input metagenome

Details	GenF size	MSL
nr-genomes size 30	30	0-45
nr-genomes size 60	60	46-80
nr-genomes size 100	100	81-125
nr-genomes size 150	150	126-175
nr-genomes size 200	200	176-225
nr-genomes size 250	250	226-275
nr-genomessize 300	300	276-300

2.2.5.2 Computing the MEBS final Entropy Score

The final Entropy Score, Sulfur Score in the case of the Sulfur cycle, can be computed by calling script *pfam_score.pl*, located in the scripts directory.This script was implicitly invoked by the score scripts presented in the "simple mode" section (2.1).

```
perl scripts/pfam_score.pl
usage: pfam_score.pl [options]
-help          Brief help message

 -input        tbl-format file with HMM matches produced by hmmsearch
               (required)

 -size         Mean peptide size length (MSL) optional,
               (default=$INP_fragment_size)
               MSL takes integers 30,60,100,150,200,250,300 for the
               analysis of metagenomes, where peptides are usually
               fragmented, and also the value 'real' in the case of
               completely sequenced genomes.

 -bzip         Input file is BZIP2-compressed  (optional)

 -entropyfile  TSV file (required) with pre-computed entropies
               from peptides of variable size, produced by
               extract_entropies.py
               The file must be formatted as follows:
```

```
                real  30  60  100 150 200 250 300
                PF00005 -0.001  0.001 ..........
                ...

-minentropy  Min relative entropy of HMMs to be considered
             to compute the Pfam score
             (optional, unsigned float example -min 0.3)

-keggmap      TSV file (optional) with HMM to KEGG mappings and
    pathway names.
              This produces a report on pathway completness and
              also a script to display the pathways in KEGG.
              The file must be formatted as follows:

                 PFAM  KO  PATHWAY   PATHWAY NAME
                 PF00890 K00394  1 Sulfite oxidation
                 PF01087   1 Sulfite oxidation
                 PF00581 K01011  2 Thiosulfate oxidation
                 ...

-pathway     Comma-separated pathway numbers from -keggmap file to
             produce pathway reports and to compute the Pfam score
             (optional, by default all pathways are used)
-random      Percent of random-sampled Pfams to compute the score
   (integer, default=100)
```

A detailed description of the optional arguments is outlined below:

1. *minentropy* This option allows you to consider a minimum entropy value to compute the final Score. For example if the user only wants to compute the Score with positive entropies, consequently discarding the negative entropies, this option can be used. For example fam domains with H'$\geqslant$ 0.3, are only considered to compute the final Score. We recommend this option only in the case that the user have previously compute the Score using default parameters. The latter is due to the importance of non-informative protein domains H'$\leq$ 0, or negative, that decreases the value of the final Score.

2. *keggmap* This option provides a detailed report of the level of completeness of the metabolic pathways of interest (See Table 2.4) and a graphical visualization using KEGG data mapping

visualization tool. In order to use this option, the user needs to provide a tabular file containing: the Pfams domains ii) the corresponding KO number, iii) the involved metabolic pathway (arbitrary number), and iv) the name of the pathway. See example file in
`sulfur_data_test/input_sulfur_data/sulfur_score_kegg_list`,

```
PFAM     KO         PATHWAY   PATHWAY NAME
PF00890  K00394      1        Sulfite oxidation
PF01087              1        Sulfite oxidation
PF00581  K01011      2        Thiosulfate oxidation
```

The first output is a detailed report of the level of completeness of each input pathway considering the number of Pfams matched in the 'omic' sample (See Table 2.4). The second output produces the list of KO numbers with Hex color codes, which correspond to the KO matches in the genomic or metagenomic data, normalized by the total number of occurrences in the total omic sample using a color scale from blue to red (being red the most abundant in the sample):

```
K11181 #d7191c,black
K15555 #fdae61,black
K02045 #fdae61,black
K10831 #fdae61,black
K16887 #fdae61,black
K17218 #abd9e9,black
K17229 #abd9e9,black
```

The output, then can be exported to the KEGG web-based interface Figure 2.11 A, and then select the metabolic map of interest. In our case we selected the metabolic map00920, to map those KOs detected in the omic sample of interest 2.11 C. The user must be aware that several KO numbers might map to the same enzymatic number.These cases are colored in KEGG map with the color of the last given KO number in the output list, and therefore the user might want to manually adjust the colors to personalize this behavior. The detailed description of the KO identifiers is also provided in KEGG 2.11 D.

Table 2.4: Pathway completeness output using the option -keggmap in the pfam_entropy.pl script

path_number	path_name	total_domains	matched	%completness	matched_Pfam_domains
1	Sulfite oxidation	9	6	66.7	PF12838...
2	Thiosulfate oxidation	10	2	20.0	PF13501...
3	Tetrathionate oxidation	2	2	100.0	PF13360...
4	Tetrathionate reduction	17	13	76.5	PF12800...
5	Sulfate reduction DS	20	15	75.0	PF00037...
6	Elemental sulfur reduction	20	10	50.0	PF13247...
7	Thiosulfate disproportion	9	7	77.8	PF12800...
8	Carbon disulfide oxidation	1	0	0.0	
9	Alkanesulfonate degradation	5	4	80.0	PF00005...
10	Sulfate reduction A	20	12	60.0	PF00005...
11	Sulfide oxidation	29	11	37.9	PF12838...
12	Cysteate oxidation	1	0	0.0	
13	Dimethylsulfone oxidation	3	2	66.7	PF01613...
14	Sulfoacetate oxidation	2	2	100.0	PF00501...
15	Sulfolactate oxidation	14	7	50.0	PF02775...
16	DMS oxidation	16	6	37.5	PF13247...
17	DMSP oxidation	12	5	41.7	PF00501...
18	MTP oxidation	7	4	57.1	PF00501..
19	Suloacetaldehyde oxidation	7	4	57.1	PF02775...
20	Elemental sulfur oxidation	7	3	42.9	PF13183...
21	Elemental sulfur disproportion	1	0	0.0	
22	Methanesulfonate oxidation	7	2	28.6	PF00111...
23	Taurine oxidation	11	6	54.5	PF00005...
24	DMS methanogenesis	2	0	0.0	
25	MTP methanogesis	2	0	0.0	
26	Methanethiol methanogenesis	2	0	0.0	
28	SQDG biosynthesis	4	3	75.0	PF00534...
29	Marker genes	12	10	83.3	PF13247...

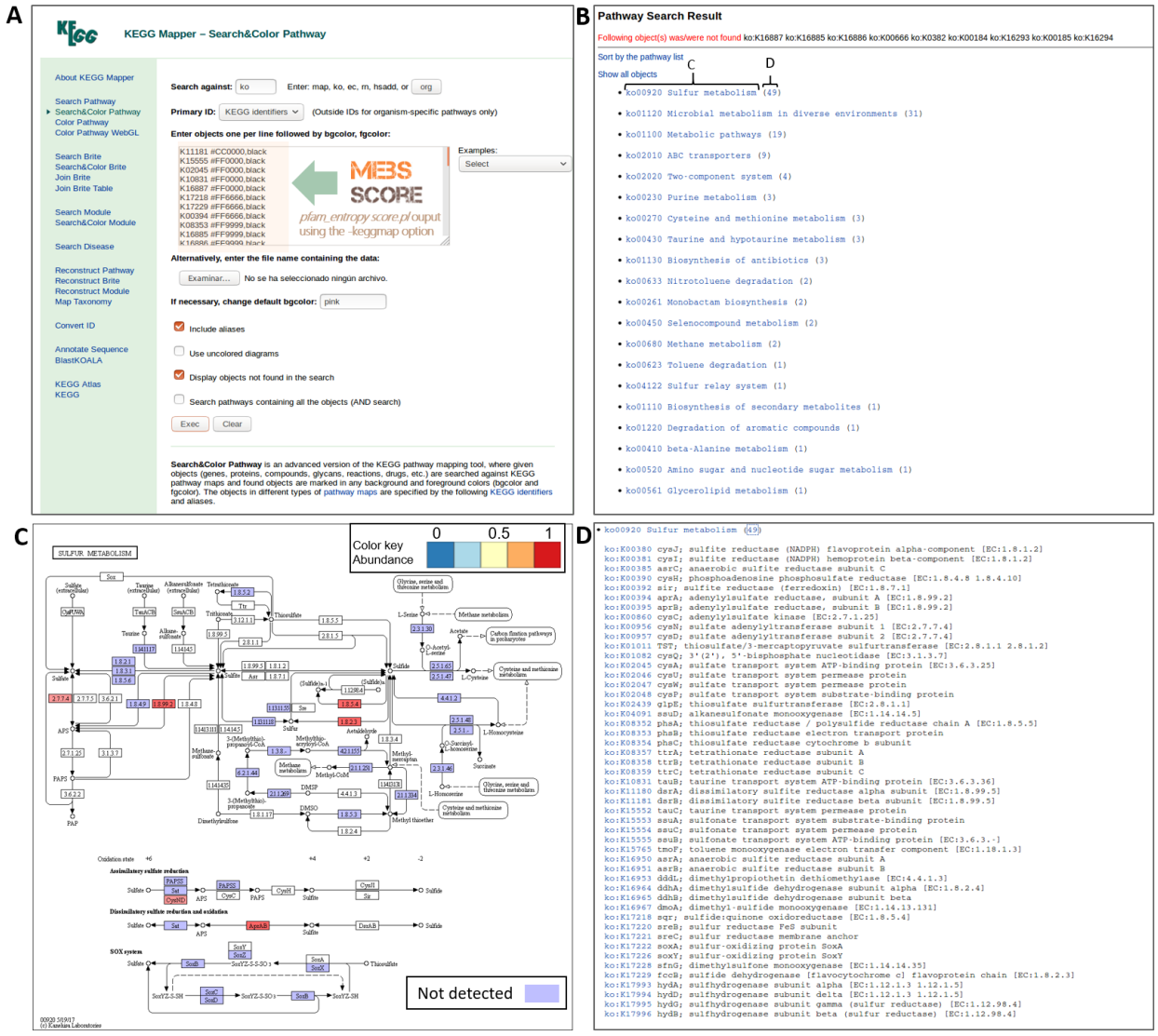

Figure 2.11: Archaeoglobus profundus DSM 5631

2.2.5.3 Some examples of how to computing MEBS final Score in Genomes

As mentioned above in section 2.1, the BASH scripts `score_genomes.sh` and `score_metagenomes.sh` read an input directory containing multiFASTA peptide sequences and compute the score using the *pfam_score.pl* script. If you want to compute the Score step by step you only need to specify the following arguments: i) the tbl format from hmmsearch see section (6), ii) the entropy file obtained in section (2.2.3) and iii) the size of your peptides, in this case *'real'*.

```
$ perl scripts/pfam_score.pl \
-in test_genomes/Enterococcus_durans.faa.out.hmmsearch.tab \
 -size real -enf sulfur_data_test/entropies_matrix_entropies.tab
```

In a more advanced mode you could also specify the option -kegg using the input file described above. `sulfur_data_test/input_sulfur_data/sulfur_score_kegg_list`,

2.2.5.4 Some examples of how to computing MEBS final Score in Metagenomes

To compute the Entropy Score in metagenomic sequences step by step, follow the below mentioned steps:

1. Run the following command to obtain the MSL of your input metagenome. Note that backslashes must be removed:

```
$ perl -lne 'if(/^(>.*)/){$h=$1}else{$fa{$h}.=$_} END{ foreach \
$h (keys(%fa)){$m+=length($fa{$h})};\
printf("MSL = %1.0f\n",$m/scalar(keys(%fa))) }'\
test_metagenomes/4440966.3_metagenome.faa >
4440966.3_metagenome.faa.msl
# MSL = 175
```

2. Using the obtained MSL, you can either choose the appropriate GenF entropies according to the Table 2.3, or you can use the following script to choose the size automatically.

```
$ perl -lne 'BEGIN{@bins=(30,60,100,150,200,250,300);\
  @th=(45,80,125,175,225,275,300)} if(/^MSL = (\S+)/){ $msl=$1; \
  foreach $i (0 .. $#th){ \
  if($msl<=$th[$i]){ print "genF = $bins[$i]"; exit } } }'\
  4440966.3_metagenome.faa.msl .msl \ >
  4440966.3_metagenome.faa.msl .genF
  cat 4440966.3_metagenome.faa.msl.genF
```

3. Save the GenF size into a variable

```
$ genF=`perl -lne 'if(/genF = (\S+)/){ print $1 }'\
  4440966.3_metagenome.faa.msl.genF`
```

4. Run the Entropy Score

```
$ perl scripts/pfam_score.pl -input \
4440966.3_metagenome.faa.out.hmmsearch.tab  \
-size $genF -enf sulfur_data_test/entropies_matrix_entropies.tab > \
4440966.3_metagenome.faa.out.hmmsearch.tab.score
```