Sets of Protein Sequences from interesting Genomes

**Blastp 🡪 Forward Best Hit 🡪 Save Data**

**Read Data 🡪 Expansion \_ Matrix 🡪 Inflation Matrix**

**Parallel\_Computation**

**Read Data🡪 Backward Best Hit 🡪 Save Data**

**Clustering the Ortholog**

ReMark : Ortholog Cluster

Matrix = 1 Blosum45

Cpu\_count = 1

Species 1,3 A ,C

Mode = 1

**List of Functions Used in Ortho\_Detection V-1\_Old**

1. **matrix\_name()**

Receive the info about matrix Number and Return the Matrix name Like "\n1. BLOSUM45\n2. BLOSUM62\n3. BLOSUM82\n4. Quit". BLOcks Substitution Matrix (BLOSUM) is a Substitution matrix used for sequence alignment of Proteins. BLOSUm45 is for more distantly related Proteins alignment Database 2. BLOSUM62: MidRange Seq with more than 62%s Similarity. 3. BLOSUM82: - More Related Proteins

1. **query\_sequence(genome):**

Receive genome Fastaq File and return gene\_sequence\_list

1. **write\_query(query, Parallel\_num)**
2. **run\_blast(subject, Parallel\_num)**
3. **same\_species\_forward\_best\_hit(blastp\_score)**

This Function return the best Score from Same species Forward best Hit. Detail about the algorithm is in Function Doc string.

1. **forward\_best\_hit(blastp\_score)**
2. **division\_parallel\_query(queryV,query\_division\_value, cpu\_count, queryV\_len)**
3. **run\_parallel\_query(species\_of\_query , species\_of\_subject , queryV, parallel\_num)**
4. **oneway\_threshold\_best\_hit(mode)**
5. **backward\_best\_hit(args)**
6. **search\_equal\_bbh\_data(target\_A)**
7. **search\_unequal\_bbh\_data(target\_B)**
8. **matching\_bbh(target)**
9. **generating\_matrix\_clustering\_ortholog(element\_set, bar)**
10. **parallel\_mcl(score\_matrix)**
11. **mcl(score\_matrix)**
12. **clustering(row\_data, col\_data, score\_matrix)**
13. **parallel\_matrix\_multiplication (data)**
14. **parallel\_matrix\_power (matrix\_element)**
15. **parallel\_matrix\_divide (data)**
16. **read\_species\_list(pr=0)** This Function is Only to display the name of Species inside the species Folder and return selected\_species\_dic (dict) , backward\_selected\_species\_dic, and total files inside species folder

(selected\_species\_dic, backward\_selected\_species\_dic, number\_i)

1. **del\_file(path, file)**
2. **check\_file(File)**
3. **read\_equal\_bbh(path)**

In this Function path is the File name where selected\_Genoem\_AA\_Genome\_bb is located.

1. **read\_unequal\_bbh(path)**
2. **Read\_species\_List()**

**Blastp\_score is output file produced by runblast()**

**Blastp\_score = RunBlast(selected\_species\_dic[Species\_of\_object, parallel\_num]**

**~~Parallel\_num is number of CPU~~**

**#Steps of Program to run**

1. **Read\_Species\_List(pr =1) Run first and return 3 values**

**## variable Name with Running Process**

1. **Mode:.** Is user input to select mode among 3 that is Blastp, Blastp Using precalculated data and Clustering.Mode 1 is time consuming so first do blastp and store all values inside blastp folder.
2. **selected\_number 🡪** Will store the user selected number which is used to extract genomes name in dictionary Format return by the function read\_species(1).

**SN** = selected\_number.split(“-“) and store selected number in a set format

**Genome\_name** selected is passed to selected\_numberand converted to sorted set and also program exit if input greater than number of Genome available

1. **Blastp\_matrix = GetMatrix() Function run and Output is “BLOSUM45” , “BLOSUM62” or “BLOSUM80”**

**,**

**BLOSUM45 is used for more distantly related alignments**

**BLOSUM62 is the Matrix built using sequences**

**BLOSUM80 is used for closely related alignment databases**

BLOSUM62 with more than 62 % Similarity (Sequences with >= 62 %identically were clustered. Note BLOSUM62 is the default matrix for protein BLAST. Experimentation has shown that the **BLOSUM-62 matrix is among the best for detecting most weak protein similarities.**

**user\_selected\_number = [backward\_selected\_species\_dic[ele] for ele in genomes]**

**If mode 1 Passed**

1. **Backward\_best\_hit\_work\_list = oneway\_Threshold\_Best\_Hit(mode)**
2. **Pool = multiprocessing.Pool(cpu\_count)**

**Description of All Variables**

1. **Mode** = User input to select Mode (1 is for Blastp. 2. Blastp using precalculated data 3. Clustering

**1** When you use the mode 2, if all of or a part of precalculated data in regard of selected genomes are not existed, the additional calculation will be conducted to make the blastp score data. Inputting the mode 3 alone, it will not worked properly. If you want to use the blastp score data which are made newly, move the file of new blastp score datea, is \*\_\*\_one\_way\_best\_hit\_\*, to the directory of blastp\_data from the directory of score\_file. And then use the mode 2, 3

1. **Selected\_species\_dic** = Dictionary value of all species inside species Folder **1**
2. **Backward\_selected\_species\_dic** = Same as 2 But the Dictionary is Opposite

**Number\_i** = length of File inside species Folder **4**

1. **blastp-score 🡪 is the blastp return value()\_**
2. **Selected\_number** = Number to select Gene like 1.AAE Gene Position in the Folder. This value is Used to take Out Species
3. **cpu\_count** = The Value of CPU For Parallel Counting (For Our Training We Only Use 1) \
4. **score\_file** = command\_options.score\_file (User input File)
5. **queryV =** GetQuerySequence(selected\_species\_dic[i]) . List format of sequence with a specific Gene
6. **queryV\_len = len(queryV)**
7. **backward\_best\_hit\_work\_list =** Oneway\_Threshold\_Best\_Hit(mode) (Return Value by the Function)
8. inflation-factor 🡪 inflation factor for clustering 1.4
9. Cluster\_out 🡪Specify the path and name to save the name of the cluster output file. The default path is current working directory and file name is result.

**name for clustering output “./cluster\_out” this is folder or Not results**

1. Species 🡪 Folder “./species/” Folder
2. Verbose 🡪 False
3. Results = pool.map(backward\_best\_hit,backward\_best\_hit\_work\_list) #This will create only after mode 1 or exists. If mode 3 only selected then this may will not exists.
4. Infinite-loop = 60
5. Log file 🡪 log file will named to name of file\_s5\_1.4.log (reverse score: 5, inflation factor :1.4)” in the mode 1 3 or 2 3

(Tip: if you want to calculate addedly new genomes with already calculated the set of genomes, use the mode2)

1. Threshold\_score 🡪 The threshold refers to a certain point in the ratings system. In order to understand the threshold, a definition of the rating system must be given. The rating is a system used to rate either ment or women and it it a score out 10.
2. Infinite\_loop 🡪 the maximum number of executions of the MCL algorithm to avoid an infinite loop during run of the MCL algorithm. The default value is “60”. 60 is the value that is strictly enforced. Recommended that you do not exceed 100.There is a case of an infinite loop calculation during the execution of MCL algorithm. In this case if the value is set too high, the calculation time increases.
3. Queue.Que() = Create a queue object with a given maximum size. If **maximum is <=0,** the queue size is infinite.default is maximum size = 0

**Important Note**

**PAM 🡪** Point Accepted Mutation

BLOSUM

**Different Between PAM & BLOSUM**

1. PAM matrices are based on an explicit evolutionary model, whereas the BLOSUM matrices are based on an implicit model of evolution.
2. The PAM matrices are based on mutations observed throughout a global alignment, this includes both highly conserved and highly mutable regions. The BLOSUM matrices are based only on highly conserved regions in series of alignments forbidden to contain gaps.
3. The method used to count the replacements is different: unlike the PAM matrix, the BLOSUM procedure uses groups of sequences within which not all mutations are counted the same.
4. Higher numbers in the PAM matrix naming scheme denote larger evolutionary distance, while larger numbers in the BLOSUM matrix naming scheme denote higher sequence similarity and therefore smaller evolutionary distance. Example: PAM150 is used for more distant sequences than PAM100; BLOSUM62 is used for closer sequences than BLOSUM50.

|  |  |
| --- | --- |
| **PAM** | **BLOSUM** |
| PAM120 | BLOSUM80(BLOSUM82 was used not availabl) |
| PAM160 | BLOSUM62 |
| PAM250 | BLOSUM45 |

**Reciprocal Best Hit algorithm (RBH)**

Reciprocal Best Hits(RBH) are a common proxy for orthology in comparative genomics.Essentially, a RBH is found when the proteins encoded by two genes, each in a different genome, find each other as the best scoring match in the other genome. NCBI’s BLAST is the software most usually used for the sequence comparisons necessary to finding RBHS.

It is used for quickly finding orthologs, i.e., genes that have diverged after a speciation event and are more likely to perform a similar function in different species, as opposed to paralogs, which are the result of a duplication event and are more likely to perform a different function. The best way to do this is by considering phylogenetic trees. If for a particular gene family (a group of genes with a similar sequence)

**Threshold\_score:🡪** Threshold\_score is the allowable range to be considered as ortholog.

If you set the Threshold score to 0 , the concept is the same as the algorithm using reciprocal best hit. The only difference is that the blastp has been performed only once.( Threshold score를 0으로 설정하면, reciprocal best hit를 이용한 알고리즘과 개념이 같게 됩니다. 다만 blastp를 한 번만 수행했다는 것이 다릅니다). Gene sequences related to reciprocal best hit are called orthologs, when the score of blastp is the best hit in both directions. In the one-way threshold best hit algorithm, the blastp is executed once to find the forward best hit (AAE Gene1 🡪 ECO Gene1) , and the data with the reverse query type gene(ECO gene1) is found in all calculated blastp datasets. After find the backward best hit. If the difference between the backward best hit score and the forward best hit score is less than or equal to the threshold score, it is considered as ‘ortholog’, and this is called the “one-way threshold best hit”

'Set the threshold score. The threshold score is an allowable range to be an ortholog. If the score difference between backward best hit score and forward best hit score is less than threshold score, it is considered the forward best hit pair to be the ortholog.' 'That is called the "one-way threshold best hit" by us. The default value is "5". "backward best hit score - forward best hit

**Protein\_sequencing :-** is the **practical process of determining the amino acid sequence** of all or part of a protein of peptide. currently the so-called protein sequencing refers to the detection of proteins primary structure, which contains the number of polypeptide chains in proteins.

The two major direct methods of protein sequencing are mass spectrometry and Edman degradation using a protein sequenator (sequencer).

The length of protein sequence is different among plant and animal (70 and 80 )

**BUGs not fixed**

1. **Glob. Glob () is currently working in windows only for mod 2**
2. Generally, Matrix 3 is causing Error
3. Do not use bare “except”
4. backward\_best\_hit\_work\_list
5. division\_parallel\_query()
6. various condition on user\_selected\_num can be reduced
7. python3.7 must be installed ( for subprocess module)

**Blastp User Manual**

**blastp** **-query** query\_file **-subject** subject\_file **-matrix** BLOSUM45 **-outfmt** 10 qseqid sseqid score length

**MCL – a Cluster algorithm for graph**

**T**he MCL algorithm is short for the Markov Cluster Algorithm, a fast and scalable unsupervised cluster algorithm for graphs (also known as networks) based on simulation of (stochastic) flow in graphs. The algorithm was invented/ discovered by Stijn Van Dongen at the Centre for Mathematics and Computer Science (also known as CWI) in the Netherlands.

**Mathematics behind the MCL Process 🡪** Expansion, which is just normal matrix multiplication, belongs to the language of linear algebra.

**Limitations of MCL**

Ofcourse , the MCL algorithm is not a panacea, and has limitations as well. Problem instances in which the diameters of the clusters are not too large allow a regime of pruning while maintaining the quality of the clusterings retrieved.

<https://micans.org/mcl/index.html?sec_thesisetc> Detail about MCL limitations can be read on the site.

\*\* concact \*\* Concat operator concatenates the output of multiple Observables so that they act like a single Observable, with all of the items emitted by the first Observable being emitted before any of the items emittexd by the second Observable (and so forth, if there are more than two)

**Distantly\_Related\_Species or**

**Function & Variable\_Name\_changed\_from\_Original\_File**

**QuerySequence 🡪 Query\_Sequence()**

**GetMatrixNumber() 🡪 MatrixName()**

**WriteQuery() 🡪 Write\_Query()**

**RunBlast() 🡪 Run\_Blast()**

**Gene\_sequence 🡪 gene\_seq**

**Gene\_sequence\_list 🡪 gene\_seq\_list**

**GetQuerySequence() 🡪QuerySequence()**

**Get\_Same\_Species\_Forward\_Best\_Hit() 🡪 Same\_Species\_Forward\_Best\_Hit()**

**GetForwardBestHit() 🡪 Forward\_Best\_Hit()**

**DivisionParallelQuery() 🡪 Division\_Parallel\_Query()**

**Parallel\_Matrix\_Multiplication\_Using\_Numpy(data) 🡪 Parallel\_Matrix\_Multiplication(data)**

**Parallel\_Matrix\_Power\_Using\_Numpy() 🡪 Parallel\_Matrix\_Power(data)**

**Parallel\_matrix\_Divide\_Using\_Numpy(data) 🡪 Parallel\_Matrix\_Divide(data)**

**Read\_Species\_List() 🡪 Read\_Species(pr=1)**

**Keys To A Good Function**

* Is Sensibly named
* Has a single responsibility
* Includes a docstring
* Returns a value
* Is no longer than 50 lines
* Is idempotent and if Possible, pure

Python PEP 8

1. Single space in operator like (a + b) not (a + b)
2. **I**nline Comments should be separated by at least two spaces from the statement.They should be separated by at least two spaces from the statement.

**Table of Abbreviations**

**BLAST 🡪 Basic Local Alignment Search Tool**

**BLOSUM 🡪 BLOck Substitution Matrix**

**COGs 🡪 The Database of Clusters of Orthologous Groups of Proteins**

**DNA 🡪 DeoxyriboNucleic Acid**

**FASTA 🡪 FAST Alignment**

**KO 🡪 Kegg Orthology**

**MCL 🡪 Markov Clustering**

**mRNA 🡪 Messaenger RiboNucleic Acid**

**NCBI 🡪 National Center for Biotechnology Information**

**OPCs 🡪 Orthologous Protein Clusters**

**PAM 🡪 Point Accepted Mutation**

**PDB 🡪 Protein Data Bank**

**RNA 🡪 RiboNucleic Acid**

**GOLD 🡪 Genome OnLine Database**

1. **Introduction:**

**It** is very significant in bioinformatics to predict the unknown functions of the other genes from the known functions of some genes by sequence homology. The BLAST algorithm is very efficient and fast, but it is very difficult to get optimal solution among distant phylogenetic species because the genomes with the large evolutionary distance typically have low similarity in their protein sequences. The orthologs from distant species likely have low BLAST scores. **Prokaryotes do not contain a cell** nucleus but **eukaryotes do it**. Prokaryotes have taxonomically two domains of bacteria and archaea. Eukaryotes consist of animals, plants, fungi, and protests.

It is necessary to understand that homology and sequence similarity are completely different. There are two **types of homologs**. One is **orthologs that are homologous genes,** which have diverged preserving the same function from each other speciation events. There are two kinds of methodology for predicting gene function based on sequence homology. One is **phylogenomic methods** and the other is **pairwise similarity-based methods.**

**The BLAST** algorithm has three steps:

* Search for exact matches of words of length W, scoring at least T, between the queryand sequences in the database.
* Try to extend the match of words that score T or greater in both directions, starting at the seed.
* Perform a gapped alilgnment between the query sequence and the database sequence using a variation of the Smith-Waterman algorithm.

**NumPy Functions used Inside owPReMark.py**

* Np.sum(score\_matrix)
* Np.concatenate(score\_matrix,divide\_results[i],axis =0)
* Numpy.matlib.ones((2,2))
* Np.ones(2,2) #need to check and compare
* Matrix.sum()
* Np.power(expansion\_matrix, float(inflation\_factor))
* Np.divide(score\_matrix, score\_matrix.sum())
* Score\_matrix\_sum = score\_matrix.sum(axis = 0)
* Score\_matrix = np.divide(score\_matrix,score\_matrix\_sum)
* Score\_matrix = numpy.matlib.zeros((len(row\_data),len(col\_data)),dtype = np.float)
* Np.zero() compare with matlib.zeros