BMS 328: Computer Science 2

Phylogenetic Trees Construction

Mayar Mansour Mohamed Ahmed

Computational Biology and Genomics, Biomedical Sciences, University of Science and Technology, Zewail City, Giza, Egypt, s-mayarmansour@zewailcity.edu.eg

Abstract

Motivation: Evolution between different species is very crucial to investigate. That is why phylogenetics, the study of evolutionary relationships among species, has been the focus of attention of many scientists. It over lots of insights in different fields for example insights regarding possible disease as pathogens can change their target host like COVID-19 now which did not infect humans this severely before, however, through evolution of strains, it became one of the greatest threats faces the world. In this project, I aim to investigate the human insulin protein to orthologs from 9 other different species: Cavia porcellus,Rattus norvegicus ,Mus musculus Mesocricetus auratus Felis catus, Canis lupus familiaris Equus caballus Gorilla and Macaca mulatta to find which species have the closest protein to the Homo sapiens using UPGMA (unweighted pair group method with arithmetic mean) method. After aligning the protein sequences of the 10 species on Multiple sequence alignment tool (Clustal Omega), I downloaded the sequence identity matrix then use it to construct the Phylogenetic tree using UPGMA based algorithm. This program was implemented using the tree data type in C++.

Results: The program showed that human and gorilla are very closely related (in the same cluster) as the rat and hamster and the horse and cat. Moreover, while comparing the algorithm with the ML and Bayesian inference algorithm, it was clear that even using the same data set, multiple trees can be inferred.

Availability: The quick brown fox jumps over the lazy dog.

Contact: s-mayarmansour@zewailcity.edu.eg

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Phylogenetics is the study of evolutionary relationships among species. These evolutionary relationships are usually inferred through the construction of phylogenetic trees shown in the figure below either based on the molecular information (genetic sequences) or the morphological characteristics (body shape, bone structure or etc.) of the species [1].

Phylogenetics is important because it enriches our understanding of how genes, genomes, species evolve. Through it, we learn how the sequences came to be the way they are today [2]. There are multiple applications of phylogenetics which include: classification as based on sequence data we can infer more accurate descriptions of patterns of relatedness than was available before molecular sequencing, Forensics as it used to assess DNA evidence presented in court cases to inform situations, e.g. where someone has committed a crime when food is contaminated, or where the father of a child is unknown, Identifying the origin of pathogens as using the phylogenetic trees and molecular data we can find out about which species the pathogen is related to and subsequently the likely source of transmission leading to new recommendations for public health policy and Conservation: as it can help to inform conservation policy when conservation biologists have to make tough decisions about which species they try to prevent from becoming extinct [3]

Insulin gene (INS) is responsible for the production of chain A and B forming the insulin hormone that control the blood glucose levels in the body [4]. It is found on chromosome 11 specifically 11p15.5 [4]. Mutations in the insulin gene can lead to Diabetes Mellitus which is a chronic disease in which the pancreas does not produce enough insulin or the body cannot effectively use the insulin produced by it [5]. In 2016, diabetes was the direct cause of 1.6 million deaths and in 2012 high blood glucose was the cause of another 2.2 million deaths [5]. Diabetic patients specifically those who have Type 1 Diabetes constantly needs to take insulin shots to regulate their blood glucose level. That is why in this project, I focused on the insulin protein to find which species that share the most homology with the homo sapiens insulin, that can easily be manipulated using biological engineering techniques to make human insulin.

UPGMA represents Unweighted Pair Group Method using Arithmetic normal. Given a distance network, it begins with gathering two taxa with the smallest distance between them [6]. After doing so, a new node is included in the midpoint of the two, and the two unique taxa are put on the tree. The new node will be connected to the others by the arithmetic mean. After that, the original network will decrease as the two taxa became one. Rehash this procedure until all taxa are put on the tree. The last taxon included will be the base of the tree [6].

In this project, I will focus on constructing a phylogenetic tree using percentage identity matrix resulted from multiple protein sequences from ten species and use trees as the appropriate data structure to do so. After downloading the protein sequences form NCBI and aligning them in Clustal Omega [7], I extracted the identity matrix and used it to build a distance based phylogenetic tree using UPGMA. Last, to investigate the efficiency, running time and compare different algorithm to construct tree, I'll compare the outputs with two platforms PhyML [8] which build the tree using maximum likelihood and MrBayes [9] which uses Bayesian inference approach.

2 Related work and Survey

There are multiple algorithms used to infer a phylogenetic tree but the most common algorithms are:

Distance algorithms which use a relevant metric of distance (Hamming distance or sequence identity) to construct a matrix showing the pair-wise distance between all organisms/genes in a dataset. This matrix is then used to construct a tree, with organisms/genes appearing on the terminal nodes, such that the distance between tips in the tree is equal to or at least close to the distance between organisms/genes in the matrix [10]. UPGMA and neighbor joining algorithm are two examples for distance algorithm

Maximum parsimony algorithms which attempt to find trees with the highest parsimony score for a given data set. If the data set is a sequence alignment, then the maximum parsimony tree is the one that allows us to produce all of the sequences in the alignment from a single ancestral sequence at the root of the said tree using the smallest number of single mutations [10]. As the maximum parsimony is always trying to find the least possible number of mutations, they are always statistically inconsistent [10].

Likelihood algorithms which either use maximum likelihood or Bayesian inference approach where both approaches involve computing the “likelihood” of phylogenies (i.e., the probability of observing your sequence alignment given that the sequences evolved along a specific tree according to a model of evolution). For the maximum likelihood approach, in spite of being moderate and computationally costly, it is the most regularly utilized phylogenetic technique especially in papers, as it is perfect for phylogeny development from sequence information [10].

3 Problem Definition

As mentioned in the introduction, Phylogenetic analysis can help us understand the origin of pathogens and how genes and proteins evolve. This project is focused on trying to understand how proteins evolve between different species, insulin hormone in particular due to its highly demand by diabetic patients. I do phylogenetic analysis and tree construction to try to find the species that has a protein closely related to the homo sapiens protein by comparing the homo sapiens INS protein with 9 other species. Using percentage identity matrix for these 10 proteins extracted from the Clustal omega multiple alignment and UPGMA, the program builds clusters that between the least distance between them which when dealing with percent identity means the ones with the highest percentage. Until all of the species are clustered together and the clusters are printed on the screen and in a txt file for further use. After clustering the species, the tree was build based on these clusters.

4 Proposed Method

In order to avoid the inconsistency with maximum parsimony algorithm and the exhaustive computational power of the likelihood algorithm, I chose to implement distance map-based algorithm. Moreover, I chose the tree as my data structure as I’m doing an evolutionary relationship analysis so it is crucial to use parents and children to infer the different clades.

4.1 The Algorithm Pseudocode & Function used:

There is a step before the coding which is getting the FASTA file from NCBI [11] then use it in [7] to get the percentage identity matrix. After preparing the file:

<u>The first step</u> in the code is declaring a new class called UPGMA that include all the function needed to do the clusters exist. When declaring the class 3 arguments are given: the name of the file containing the lower half matrix as it is symmetrical, and the number of rows and columns which are equivalent to the number of species you are investigating +1. The UPGMA class automatically calls for creatematrix() and loadmatrix() functions

<u>The second step</u> is including 5 functions within the UPGMA class for clustering the species: creatematrix(), loadmatrix(), printmatrix(), minimum_value() and finally Build_Cluster().

The creatematrix() function creates a zero matrix with the dimensions given when declaring a UPGMA class.

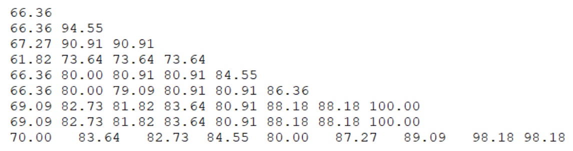
The function loadmatrix() read the file containing the identity matrix and put it in the zero matrix created by creatmatrix() and making sure that m[i,x]= m[x,i]

Printmatrix() function prints the matrix resulted after each cluster

Minimum_value(): finds the value with the highest percentage identity (i.e. the value where 2 species are closest to each other) and print this value , its position and the 2 taxa it was found between

Build_Cluster():uses the positions extracted from minimum_value function to build the cluster by taking the average between them and other pints in the matrix then call the printing function to print the new matrix. it keeps redoing that until the all elements are clustered and the last matrix is size 2x2. Moreover, every cluster and the distance between them are saved in a file called tree.txt

<u>The third Step</u> is using the tree.txt file to build the tree which includes building a class named node and a print function called printTree() to show the resulted tree.



```
66.36
66.36 94.55
67.27 90.91 90.91
61.82 73.64 73.64 73.64
66.36 80.00 80.91 80.91 84.55
66.36 80.00 79.09 80.91 80.91 86.36
69.09 82.73 81.82 83.64 80.91 88.18 88.18 100.00
69.09 82.73 81.82 83.64 80.91 88.18 88.18 100.00
70.00   83.64   82.73  84.55  80.00   87.27   89.09   98.18 98.18
```

Fig.2: The percentage identity matrix resulted from the Clustal Omega

4.2 Algorithm complexity analysis

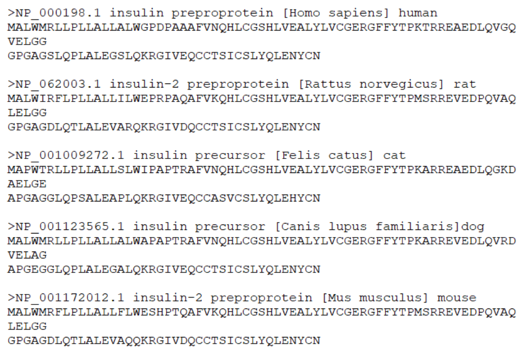
For Clustering the species using UPGMA:
Creatematrix() had O(N^2) as it has a nested loop with each of them N so their multiplication will be O(N^2)
Loadmatrix() had O(N^2) as it has a nested loop with each of them N so their multiplication will be O(N^2)
Printmatrix() has O($N+N^2$)=O(N^2) as it has a nested loop so N^2 and another loop so their addition will lead to O(N^2)
Minmum_value () has O($N+N^2$) = O(N^2) as it has a nested loop so N^2 and another loop so their addition will lead to O(N^2)
Build_cluster() has O($N+N^2 +N^3$)=O(N^3) as it has a nested loop so N^2 ,another nested loop inside a loop so O(N^3) and a single loop= O(N) so their addition will lead to O(N^3) where n is the number of columns =number of rows as it is a square matrix

For the tree construction:
The print function has a loop so it will be O(H) and the insertion was hard coded leading to an O(H^2) so their total is O($H+H^2$) =O(H^2) where h is the height of the tree

5 Results/Evaluation/Experiments

5.1 Data Details of Data set utilized

The data used in this project was INS protein sequence extracted from the protein database in NCBI for 10 species: Cavia porcellus,Rattus norvegicus ,Mus musculus Mesocricetus auratus Felis catus, Canis lupus familiaris, Equus caballus, Gorilla and Macaca mulatta as a FASTA file then aligned in Clustal omega to get the precentage identity matrix in a text file. The file containing the matrix only contained the lower bound to avoid redundancy as it is a symmetrical matrix. Sample from the FASTA file and the matrix are shown in figure 1 and 2 respectively.



```
>NP_000198.1 insulin preproprotein [Homo sapiens] human
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQ
VELGG
GPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN

>NP_062003.1 insulin-2 preproprotein [Rattus norvegicus] rat
MALWIRFLPLLALLILWEPRPAQAFVKQHLCGSHLVEALYLVCGERGFFYTPMSRREVEDPQVAQ
LELGG
GPGAGDLQTLALEVARQKRGIVDQCCTSICSLYQLENYCN

>NP_001009272.1 insulin precursor [Felis catus] cat
MAPWTRLLPLLALLSLWIPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREAEDLQGKD
AELGE
APGAGGLQPSALEAPLQKRGIVEQCCASVCSLYQLEHYCN

>NP_001123565.1 insulin precursor [Canis lupus familiaris]dog
MALWMRLLPLLALLALWAPAPTRAFVNQHLCGSHLVEALYLVCGERGFFYTPKARREVEDLQVRD
VELAG
APGEGGLQPLALEGALQKRGIVEQCCTSICSLYQLENYCN

>NP_001172012.1 insulin-2 preproprotein [Mus musculus] mouse
MALWMRFLPLLALLFLWESHPTQAFVKQHLCGSHLVEALYLVCGERGFFYTPMSRREVEDPQVAQ
LELGG
GPGAGDLQTLALEVAQQKRGIVDQCCTSICSLYQLENYCN
```

Fig.1: Sample from the FASTA file

5.2 The List of questions the experiments are designed to answer

The project was investigating the evolution of INS protein among 10 species (Cavia porcellus, Rattus norvegicus, Mus musculus, Mesocricetus auratus, Felis catus, Canis lupus familiaris, Equus caballus, Gorilla and Macaca_mulatta) to answer the following

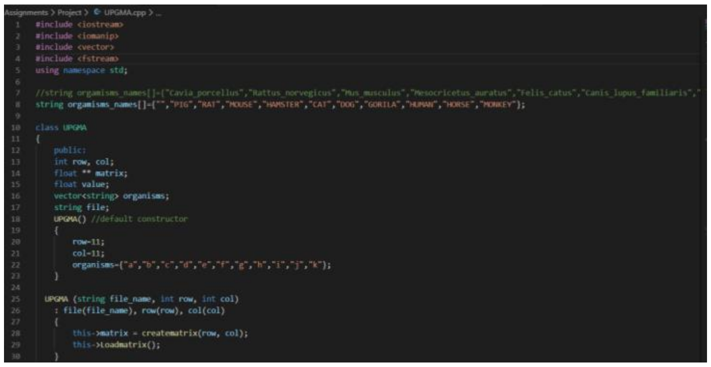
(1) Is the INS gene highly conserved between species?
(2) Can we construct a phylogenetic tree to give us an idea about which species have the closest homology with the homo sapiens?
(3) Which algorithm from the three most commonly used ones in constructing the phylogenetic tree can achieve balance between complexity, speed and accuracy?
(4) Can we say that one phylogenetic tree is the correct one?

5.3 Observations and the Interpretations

The coding pipeline of the project can be divided into 3 parts:1- loading the matrix and building the UPGMA class, 2- Finding the closest species and Building the clusters then saving them in a txt file 3- Constructing the phylogenetic tree from the clusters build from UPGMA by implementing tree data structure.

5.3.1 Loading the matrix from the file and building the UPGMA class

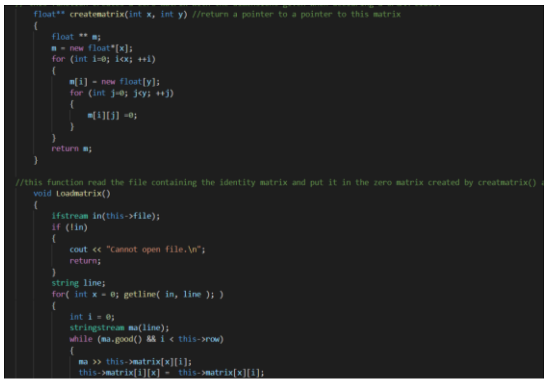
This step include declaring the UPGMA class and building the loadmatrix(), printmatrix()and the creatematrix() functions. The code for this part is shown in figure 3 and 4. Moreover, as the matrix file does not include the header indicating to which organism each percentange belong to, the header we saved in a string array called organims_names() in order related to the matrix.



Assignments > Project > UPGMA.cpp > ...

```
#include <iostream>
#include <iomanip>
#include <vector>
#include <fstream>
using namespace std;

//string orgamisms_names[]={"Cavia_porcellus","Rattus_norvegicus","Mus_musculus","Mesocricetus_auratus","Felis_catus","Canis_lupus_familiaris","
string orgamisms_names[]={"","PIG","RAT","MOUSE","HAMSTER","CAT","DOG","GORILA","HUMAN","HORSE","MONKEY"};

class UPGMA
{
    public:
    int row, col;
    float ** matrix;
    float value;
    vector<string> organisms;
    string file;
    UPGMA() //default constructor
    {
        row=11;
        col=11;
        organisms={"a","b","c","d","e","f","g","h","i","j","k"};
    }

  UPGMA (string file_name, int row, int col)
    : file(file_name), row(row), col(col)
    {
        this->matrix = creatematrix(row, col);
        this->Loadmatrix();
    }
```

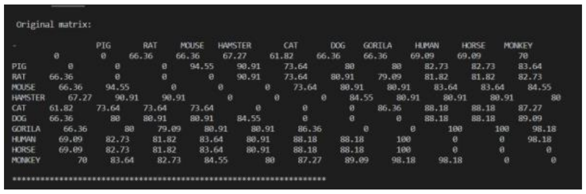
Fig.3: Coding part for the UPGMA class



```
float** creatematrix(int x, int y) //return a pointer to a pointer to this matrix
{
    float ** m;
    m = new float*[x];
    for (int i=0; i<x; ++i)
    {
        m[i] = new float[y];
        for (int j=0; j<y; ++j)
        {
            m[i][j] =0;
        }
    }
    return m;
}

//this function read the file containing the identity matrix and put it in the zero matrix created by creatmatrix()
void Loadmatrix()
{
    ifstream in(this->file);
    if (!in)
    {
        cout << "Cannot open file.\n";
        return;
    }
    string line;
    for( int x = 0; getline( in, line ); )
    {
        int i = 0;
        stringstream ma(line);
        while (ma.good() && i < this->row)
        {
          ma >> this->matrix[x][i];
          this->matrix[i][x] =  this->matrix[x][i];
```

Fig.4: Shows part of loadmarix() and creatematix function.

The results from this part shows that the code was successfully be to read and parse the txt file containing the matrix. Moreover, it recreated the full matrix from just the lower part provided in the file. The original matrix is shown in figure 5.

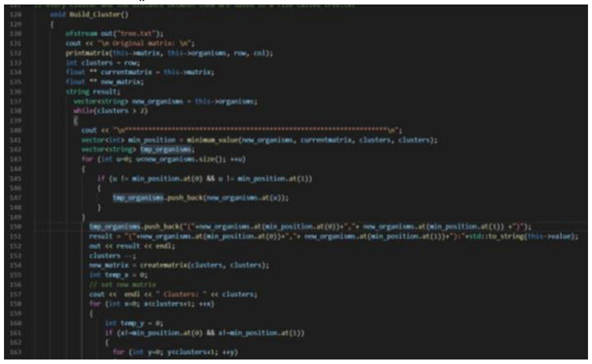


```
Original matrix:

-                  PIG       RAT      MOUSE   HAMSTER      CAT       DOG     GORILA     HUMAN     HORSE    MONKEY
          0          0     66.36     66.36     67.27     61.82     66.36     66.36     69.09     69.09        70
PIG          0         0         0     94.55     90.91     73.64        80        80     82.73     82.73     83.64
RAT      66.36         0         0         0     90.91     73.64     80.91     79.09     81.82     81.82     82.73
MOUSE      66.36     94.55         0         0         0     73.64     80.91     80.91     83.64     83.64     84.55
HAMSTER      67.27     90.91     90.91         0         0         0     84.55     80.91     80.91     80.91        80
CAT      61.82     73.64     73.64     73.64         0         0         0     86.36     88.18     88.18     87.27
DOG      66.36        80     80.91     80.91     84.55         0         0         0     88.18     88.18     89.09
GORILA      66.36        80     79.09     80.91     80.91     86.36         0         0       100       100     98.18
HUMAN      69.09     82.73     81.82     83.64     80.91     88.18     88.18       100         0         0     98.18
HORSE      69.09     82.73     81.82     83.64     80.91     88.18     88.18       100         0         0         0
MONKEY         70     83.64     82.73     84.55        80     87.27     89.09     98.18     98.18         0         0

*******************************************************************
```

Fig.5: Shows the original matrix completed after parsing the file

5.3.2 Finding the closest species and Building the clusters then saving them in a new txt file

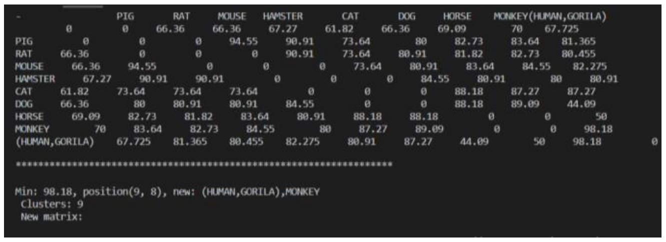
After retrieving the original matrix shown in figure 5, the main function calls for the Build_Cluster() function which respectively call for minimum_value() function to find the closest 2 taxa and combining them into a cluster. After finding the minimum value which mentioned before is the highest percent identity, Build_Cluster() function reevaluate the matrix to form a new one with the 2 taxa combined and the distance between them and other taxa is modified. The new matrix is printed in the terminal along with the position of the minimum value and its value. Build_cluster() keep reevaluating the matrix till all the species are clustered together and a file named tree.txt is formed to contain all the clusters. Figure 6 shows part of the coding for the Build_cluster() function.



```
void Build_Cluster()
{
    ofstream out("tree.txt");
    cout << "\n Original matrix: \n";
    printmatrix(this->matrix, this->organisms, row, col);
    int clusters = row;
    float ** currentmatrix = this->matrix;
    float ** new_matrix;
    string result;
      vector<string> new_organisms = this->organisms;
      while(clusters > 2)
      {
        cout << "\n*******************************************************************\n";
        vector<int> min_position = minimum_value(new_organisms, currentmatrix, clusters, clusters);
        vector<string> tmp_organisms;
        for (int u=0; u<new_organisms.size(); ++u)
        {
            if (u != min_position.at(0) && u != min_position.at(1))
            {
                tmp_organisms.push_back(new_organisms.at(u));
            }
        }
          tmp_organisms.push_back("("+new_organisms.at(min_position.at(0))+","+ new_organisms.at(min_position.at(1)) +")");
          result = "("+new_organisms.at(min_position.at(0))+","+ new_organisms.at(min_position.at(1))+"):"+std::to_string(this->value);
          out << result << endl;
          clusters --;
          new_matrix = creatematrix(clusters, clusters);
          int temp_x = 0;
          // set new matrix
          cout <<  endl << " Clusters: " << clusters;
          for (int x=0; x<clusters+1; ++x)
          {
              int temp_y = 0;
              if (x!=min_position.at(0) && x!=min_position.at(1))
              {
                for (int y=0; y<clusters+1; ++y)
```

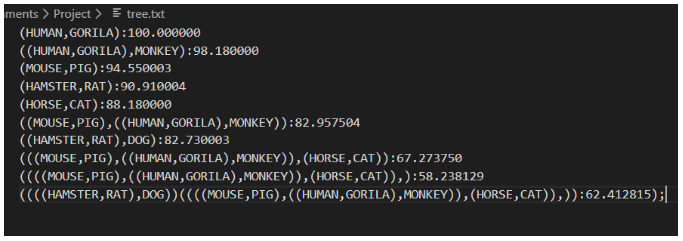
Fig.6: Shows part of the coding for Build_Cluster() function

The new matrix after finding the first cluster is shown in figure 7. It is clear that human and gorilla were made into a cluster that is closely related to monkeys. Morover, after the hash symbols it clearly states that the next taxa will be monkey which will be clustered with the (human, gorilla) cluster created earlier. Moreover, the tree.txt file updated each time a cluster form contains all the cluster along with the distance between them. It is shown in figure 8.



```
-                  PIG       RAT      MOUSE   HAMSTER      CAT       DOG     HORSE    MONKEY(HUMAN,GORILA)
          0          0     66.36     66.36     67.27     61.82     66.36     69.09        70    67.725
PIG          0         0         0     94.55     90.91     73.64        80     82.73     83.64    81.365
RAT      66.36         0         0         0     90.91     73.64     80.91     81.82     82.73    80.455
MOUSE      66.36     94.55         0         0         0     73.64     80.91     83.64     84.55    82.275
HAMSTER      67.27     90.91     90.91         0         0         0     84.55     80.91        80     80.91
CAT      61.82     73.64     73.64     73.64         0         0         0     88.18     87.27     87.27
DOG      66.36        80     80.91     80.91     84.55         0         0     88.18     89.09     44.09
HORSE      69.09     82.73     81.82     83.64     80.91     88.18     88.18         0         0        50
MONKEY         70     83.64     82.73     84.55        80     87.27     89.09         0         0     98.18
(HUMAN,GORILA)    67.725    81.365    80.455    82.275     80.91     87.27     44.09        50     98.18         0

*******************************************************************

Min: 98.18, position(9, 8), new: (HUMAN,GORILA),MONKEY
 Clusters: 9
 New matrix:
```

Fig.7: Shows first reevaluated matrix after finding the first cluter which is human and gorilla



ments > Project > ≡ tree.txt

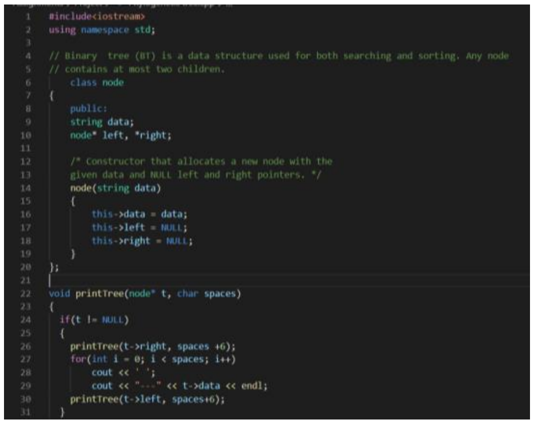
```
(HUMAN,GORILA):100.000000
((HUMAN,GORILA),MONKEY):98.180000
(MOUSE,PIG):94.550003
(HAMSTER,RAT):90.910004
(HORSE,CAT):88.180000
((MOUSE,PIG),((HUMAN,GORILA),MONKEY)):82.957504
((HAMSTER,RAT),DOG):82.730003
(((MOUSE,PIG),((HUMAN,GORILA),MONKEY)),(HORSE,CAT)):67.273750
((((MOUSE,PIG),((HUMAN,GORILA),MONKEY)),(HORSE,CAT)),):58.238129
((((HAMSTER,RAT),DOG))((((MOUSE,PIG),((HUMAN,GORILA),MONKEY)),(HORSE,CAT)),)):62.412815);
```

Fig.8: Shows the tree.txt file with all the clusters and distance between them

The most interesting fact about the clusters and the distance between them is that human and gorilla are 100% identical meaning that according to this tree gorilla insulin hormone can be used as a homo sapiens hormone with minimal bioengineering process.

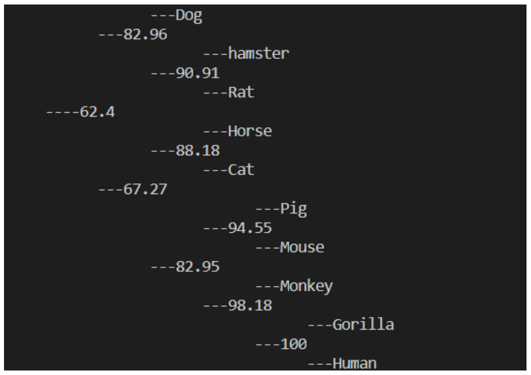
5.3.3 Constructing the phylogenetic tree from the clusters build from UPGMA by implementing tree data structure

Using the clusters in the tree.txt file, class node was created based on tree data structure implementation was built containing a function to create nodes and a print function to print the resulted tree. The tree was built with the distance between each cluster as its node and the taxa as the leaves. The coding part is shown in figure 9.



```
#include<iostream>
using namespace std;

// Binary  tree (BT) is a data structure used for both searching and sorting. Any node
// contains at most two children.
    class node
{
    public:
    string data;
    node* left, *right;

    /* Constructor that allocates a new node with the
    given data and NULL left and right pointers. */
    node(string data)
    {
        this->data = data;
        this->left = NULL;
        this->right = NULL;
    }
};

void printTree(node* t, char spaces)
{
  if(t != NULL)
  {
    printTree(t->right, spaces +6);
    for(int i = 0; i < spaces; i++)
        cout << ' ';
        cout << "---" << t->data << endl;
    printTree(t->left, spaces+6);
  }
```

Fig.9: Shows coding for the Node clase for the tree construction



```
            ---Dog
      ---82.96
                  ---hamster
            ---90.91
                  ---Rat
----62.4
                  ---Horse
            ---88.18
                  ---Cat
      ---67.27
                        ---Pig
                  ---94.55
                        ---Mouse
            ---82.95
                        ---Monkey
                  ---98.18
                              ---Gorilla
                        ---100
                              ---Human
```

Fig.10: Shows the tree resulted form the clustering data in tree.txt

Figure 10 clearly shows that human and gorilla are in the same clade which conside with the other algorithms. This tree also agrees with the ML tree on the monkey being the closest to the gorilla and human clade. It also shows that the horse and cat are in a clade together and the hamster and rat are together

5.4 Problems and assumptions in the code

The program is built on the assumption that all sequences have equal length in order to avoid insertion and deletion. Moreover, the header for the matrix is built on the assumption that the user will use the same data set as me, but this can be easily changed by modifying the organism_names array. The UPGMA is built on the assumption that all mutations are equally plausible and that there is a constant molecular clock.

There were many issues facing me during coding the build_cluster () function regarding how to keep track of the organisms and the distance, how to update the matrix and so on. There were also issues with how to use the data in the string to built the tree and I could not figure a way to parse it properly so I hard code it.

5.5 Comparison with another tools/ algorithms

As mentioned earlier, this algorithm was compared with 2 others: maximum likelihood implemented in PhyML and Bayesian inference approach implemented in MrBayes. Both platforms form a multiple sequence alignment before forming the matrix and constructing the tree.

For the PhyML, it implements maximum likelihood (ML) algorithm but with some adjustment to make minimize the extensive computational power used by most of the other ML programs as it starts by building an initial tree using fast distance-based method then modifying it to improve its likelihood at each iteration [12]. Figure 11 shows the tree resulted from it and it clearly shows that human and Gorilla are in the same clade which is the same conclusion reached by this program.

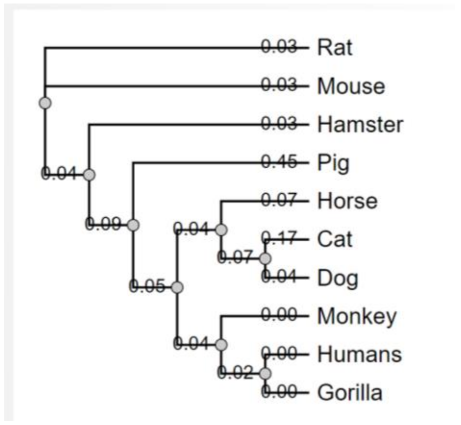
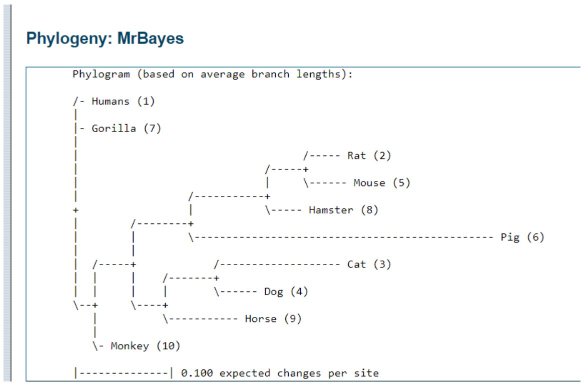


Fig.11: Show the tree resulted from PhyML (Maximum Liklihood Algorithm)

As for the Bayesian inference approach implementation in MrBayes, the phylogenetic analysis is based on posterior probabilities of phylogenetic trees that are build using Markov Chain Monte Carlo (MCMC) [13]. Bayesian inference tree as seen in figure 12 did not make the human and gorilla in the same clade as the UPMGA and ML, however, it did put them very close to each other.



Phylogeny: MrBayes

```
Phylogram (based on average branch lengths):

/- Humans (1)
|
|- Gorilla (7)
|
|                                   /----- Rat (2)
|                             /-----+
|                             |     \------ Mouse (5)
|                 /-----------+
+                 |           \----- Hamster (8)
|        /--------+
|        |        \----------------------------------------------- Pig (6)
|        |
|  /-----+            /------------------- Cat (3)
|  |     |    /-------+
|  |     |    |       \------ Dog (4)
\--+     \----+
   |          \----------- Horse (9)
   |
   \- Monkey (10)

|--------------| 0.100 expected changes per site
```

Fig.12: Show the tree resulted from MrBayes

Comparing the trees resulted from each algorithm shows that there are multiple trees that can be inferred from the same data set. No one can say that one of them is surely wrong as there is no complete data showing all the ancestors. Phylogenetic trees merely infer that evolutionary relationships between different species. There are multiple algorithms that can construct them and each of them can construct different trees. Despite that there are somethings that are fairly constant throughout multiple trees which in the data set used here is how closely related the insulin hormone between the Homo sapiens and the Gorilla.

6. Conclusion

In conclusion, the program works efficiently and very user friendly. It can parse the txt file entered by the containing the lower half of a percentage identity matrix to form clusters between the different species and build a phylogenetic tree. Moreover, it saves the different clusters in a txt file for further usage latter on. While investigating the insulin protein between 10 different species, the data showed that the human and gorilla are closely related to one another. Moreover, comparing different algorithm using the same data set showed that there are multiple possible trees to be constructed from the same data set not just one.

7. Suggestion and Improvement

To further test the evolutionary relationship between the species, it might be best if we try using sequences with unequal length, introducing insertion and deletion. Moreover, we can introduce mutation models like Kimura 2-parameter (K2P) which assume that transitions more likely than transversions.

Acknowledgements

This was done under the supervision of Dr Eman Mostafa. I would like to use this opportunity to thank Dr Eman for her effort and guidance throughout the semester along with Mr. Mohamed Kerdawy.

Funding

This work was not funded by anyone

Conflict of Interest: none declared.

References

[1] Zhang, A., 2019. Understanding Genetics. [online] Genetics.thetech.org. Available at: <https://genetics.thetech.org/ask-a-geneticist/how-build-phylogenetic-tree>

[2] Yuan, J., Zhu, Q. and Liu, B., 2014. Phylogenetic and Biological Significance of Evolutionary Elements from Metazoan Mitochondrial Genomes. PLoS ONE, 9(1),p.e84330.

[3] EMBL-EBI Train online. n.d. Why Is Phylogenetics Important? [online] Available at:<https://www.ebi.ac.uk/training/online/course/introduction-phylogenetics/why-phylogenetics-important>

[4] G. Reference, "INS gene", Genetics Home Reference. [Online]. Available: https://ghr.nlm.nih.gov/gene/INS#location.

[5] "Diabetes", Who.int, 2016. [Online]. Available: https://www.who.int/newsroom/fact-sheets/detail/diabetes.

[6] H. Singh, E. Singh and N. Kaur, "Implementing Hierarchical Clustering Method for Multiple Sequence Alignment and Phylogenetic Tree Construction", International Journal of Computer Science, Engineering and Information Technology, vol. 3, no. 1, pp. 1-12, 2013. Available: 10.5121/ijcseit.2013.3101.

[7] H. McWilliam et al., "Analysis Tool Web Services from the EMBL-EBI", Nucleic Acids Research, vol. 41, no. 1, pp. W597-W600, 2013. Available: 10.1093/nar/gkt376

[8] F. Lemoine et al., "NGPhylogeny.fr: new generation phylogenetic services for non-specialists", Nucleic Acids Research, vol. 47, no. 1, pp. W260-W265, 2019. Available: 10.1093/nar/gkz303 [Accessed 3 June 2020].

[9] A. Dereeper, S. Audic, J. Claverie and G. Blanc, "BLAST-EXPLORER helps you building datasets for phylogenetic analysis", BMC Evolutionary Biology, vol. 10, no. 1, p. 8, 2010. Available: 10.1186/1471-2148-10-8.

[10] "A brief guide to the different methods of phylogenetic tree construction | ZAGENO", ZAGENO. [Online]. Available: https://zageno.com/l/guide-to-different-methods-of-phylogenetic-tree-construction?utm_medium=article&utm_campaign=r_phylogenetic_tree_construction.

[11] https://www.ncbi.nlm.nih.gov/protein/?term=ins

[12] S. Guindon, F. Lethiec, P. Duroux and O. Gascuel, "PHYML Online--a web server for fast maximum likelihood-based phylogenetic inference", 2005. [Online]. Available: https://academic.oup.com/nar/article/33/suppl_2/W557/2505422.