Algorithms for Bioinformatics

2018/2019

Phylogenetic Analysis

Pedro G. Ferreira
[d**CC**] @ Faculty of Sciences University of Porto

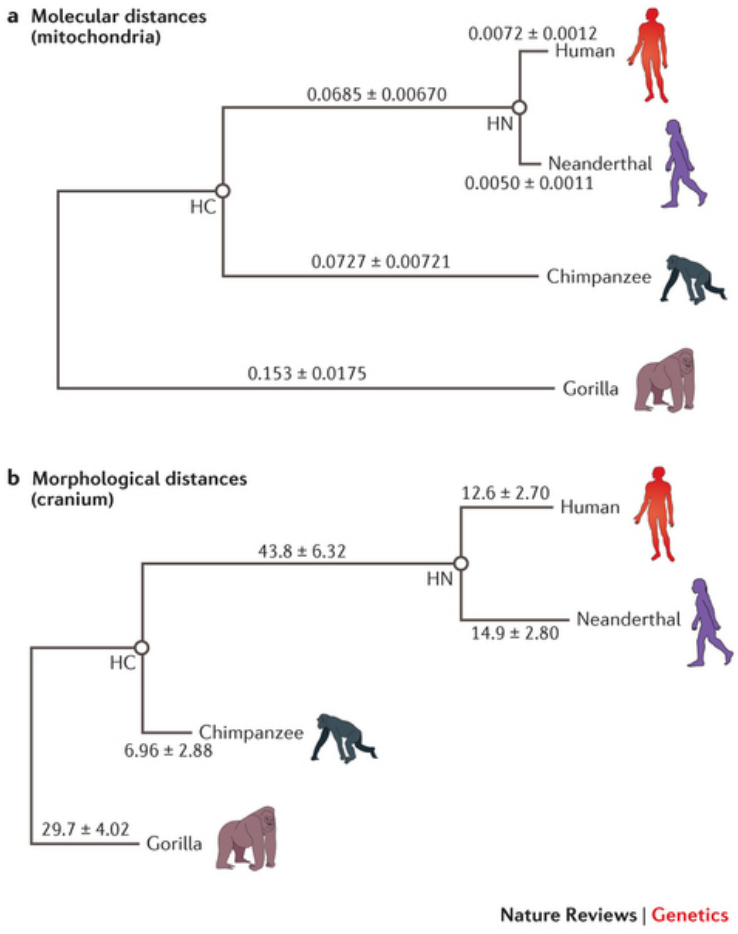
Outline

- Phylogenetic Trees: problem definition and relevance
- Classes of algorithms for Phylogenetic Analysis
- Hierarchical Clustering
- Unweighted Pair Group Using Arithmetic Averages (UPGMA)
- Implementing distance-based algorithms in Python
- Implementing Binary Trees
- Implementing the UPGMA
- Exercises

Phylogenetic Analysis

- Phylogenetics studies the evolutionary history and relationship among individuals or species.
- Phylogenetics trees illustrate the relationship between these individuals.
- Phylogenetics analysis based on molecular data are more rigorous than those based on heritable traits (e.g. morphological measurements).

Phylogenetics



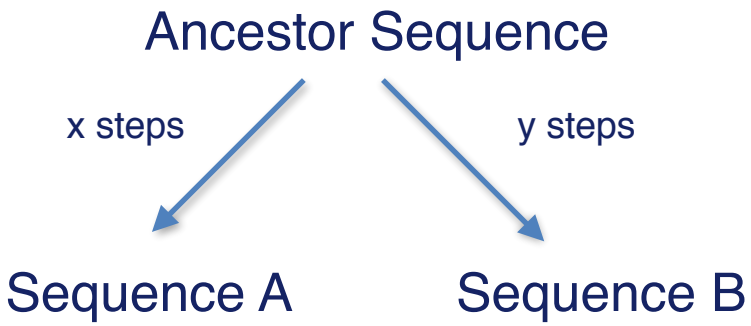
Depending on the nature of the characters different evolutionary rates can be observed:

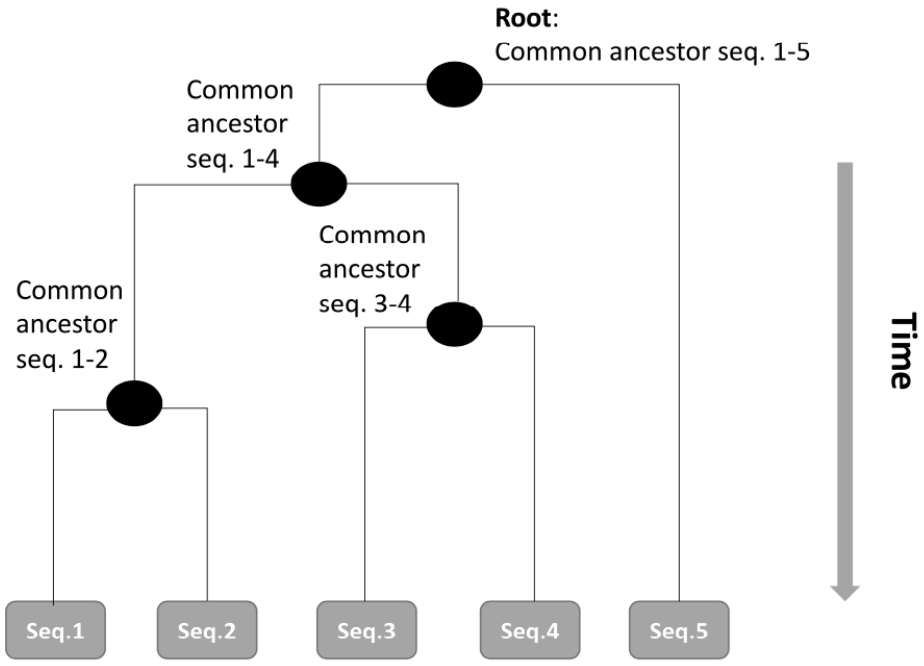
- Molecular sequences have a nearly constant rate in these close species.
- Morphological characteristics evolve in a different way.

Bayesian molecular clock dating of species divergences in the genomics era
Mario dos Reis, Philip C. J. Donoghue & Ziheng Yang
Nature Reviews Genetics 17, 71–80 (2016) doi:10.1038/nrg.2015.8

Phylogenetic tree

- Phylogenetics trees:
  - Leaves are sequences (typically from different species or taxonomic categories)
  - Internal nodes represent common ancestors of the sequences.
  - The structure of a rooted tree may be represented by clusters.
  - The height of the nodes in the tree represent a measure of time (moving from the root to the leaves).
  - Unrooted trees illustrate the relation between the leaves without explicitly inferring the common ancestor.





- From Bioinformatics Algorithms, Rocha & Ferreira

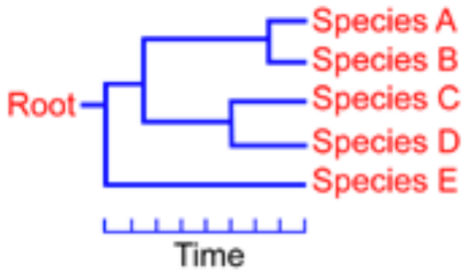
Phylogenetic tree

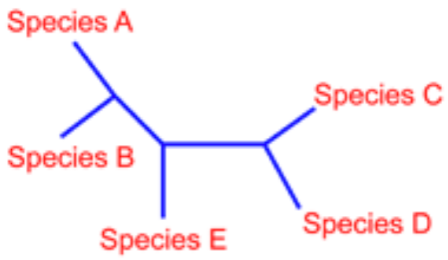
- Trees may contain a root node; nodes represent the ancestors; the leafs represent the sequences under analysis; branches represent the evolutionary distance between sequences.

- The length of the branch represents the evolutionary distance between the ancestor and the species at the node. This is captured by the number of sequence changes between one level and the next level of the tree.

- **Rooted tree**: contains a common ancestor that is the latest ancestor. The direction of the path from the root to any other node in the tree indicates the passage of time.

- **Unrooted tree**: there is no hierarchy imposed by a main node (oldest common ancestor).





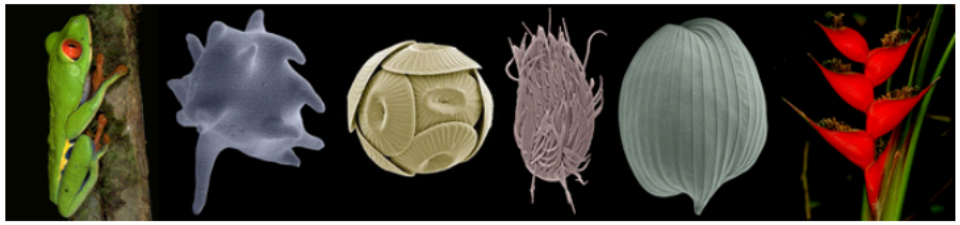
- Credit to Enrique Blanco; Taken from http://genome.crg.es/courses/msa/

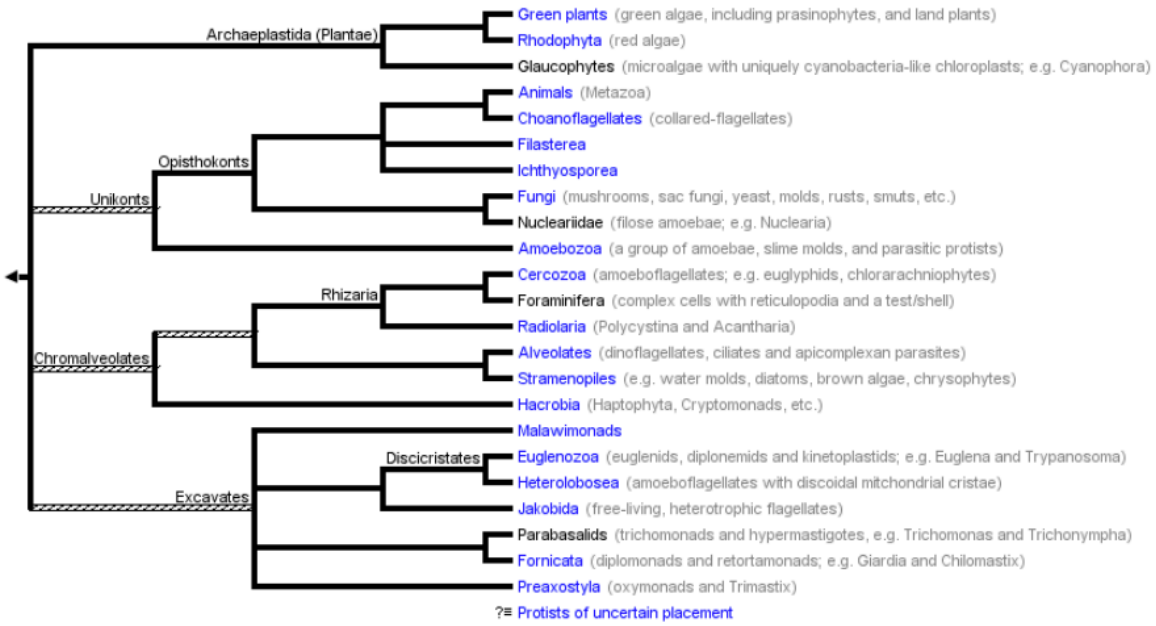
The tree of life: tolweb.org

Eukaryotes

Eukaryota, Organisms with nucleated cells

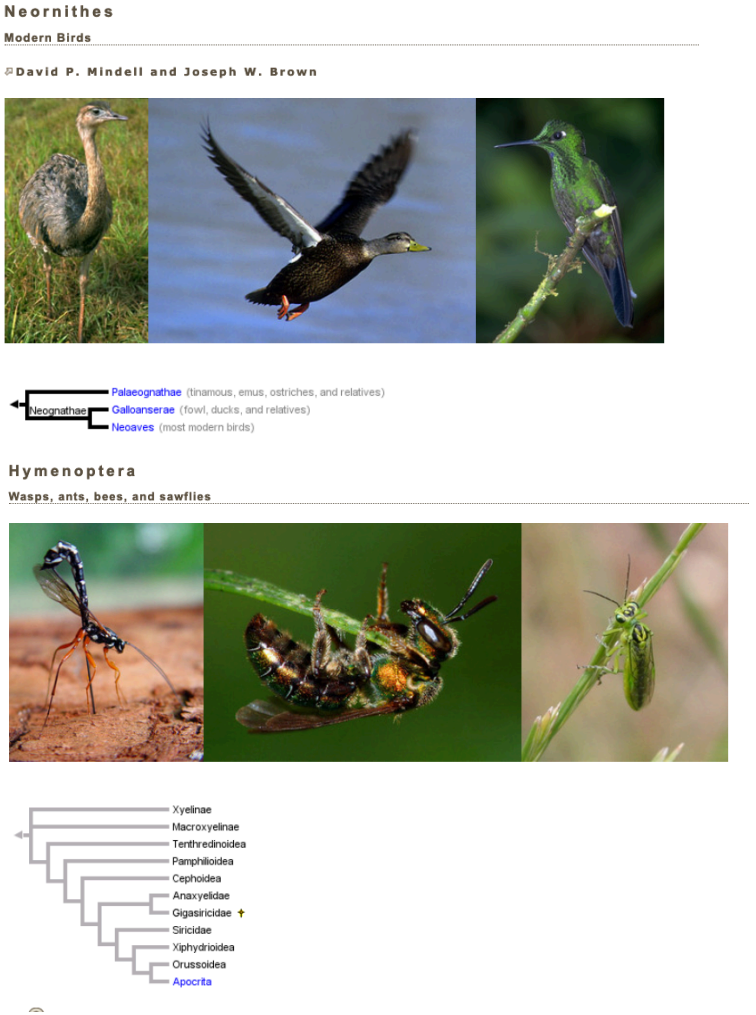
Patrick Keeling, Brian S. Leander, and Alastair Simpson





The tree of life

- Eubacteria
- **Eukaryotes**
  - **Animals**
    - Echinoderms (sea urchins, starfish, sea cucumbers, etc)
    - **Vertebrates (fish etc.)**
      - **Terrestrial Vertebrates**
        - Frogs
        - Salamanders
        - Turtles
        - **Dinosaurs**
          - Modern Birds
        - Mammals
      - Teleost fish
    - Cnidaria (jellyfish, anemones, corals, etc.)
    - Annelida (segmented worms)
    - Cephalopoda (octopods, squids, etc.)
    - **Arthropoda**
      - **Insects**
        - Dragonflies and Damselflies
        - Lice
        - True Bugs
        - Beetles
        - Wasps, Bees, and Ants
        - Flies
        - Butterflies and Moths
        - Crickets, Katydids, and Grasshoppers
      - **Arachnids**
        - Spiders
        - Mites
        - Scorpions
  - Fungi
  - **Green Plants**
    - Ferns
    - Flowering Plants

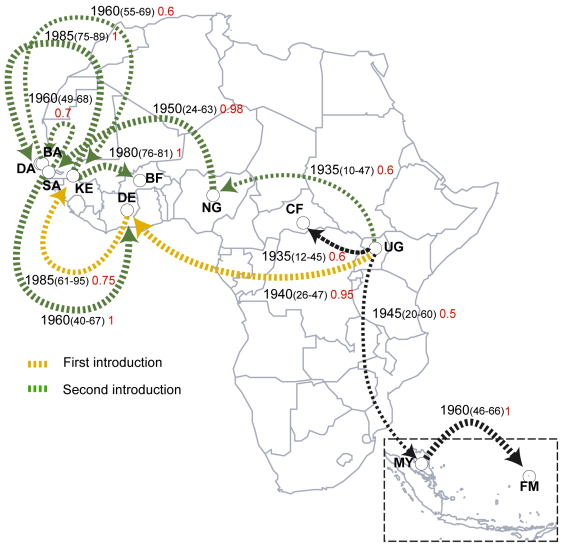


- Phylogenetics studies help to classify new species and provide rigorous support for the definition of taxonomic categories.

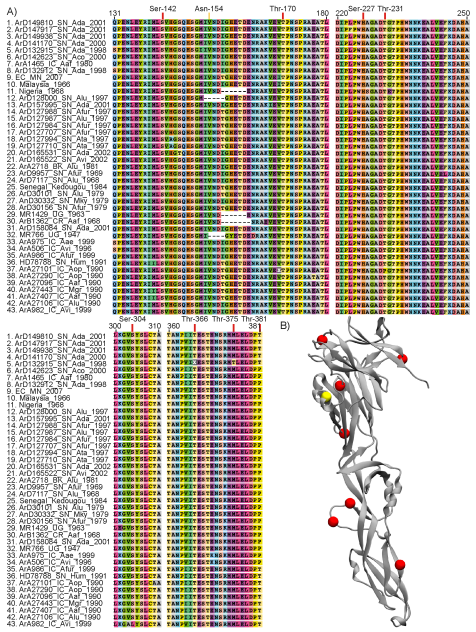
Virus Phylogenies

- 37 ZIKV isolates collected from 1968 to 2002 in six localities in Senegal and Côte d'Ivoire.
- The ZIKV genome consists of a single-stranded positive sense RNA molecule with 10794 kb.
- Faye O. et al. Molecular Evolution of Zika Virus during Its Emergence in the 20th Century. PLoS Negl Trop Dis 8(1): (2014).

Maximum likelihood phylogenetic tree inferred for concatenated of sequences from Envelope and NS5 genes of Zika virus.



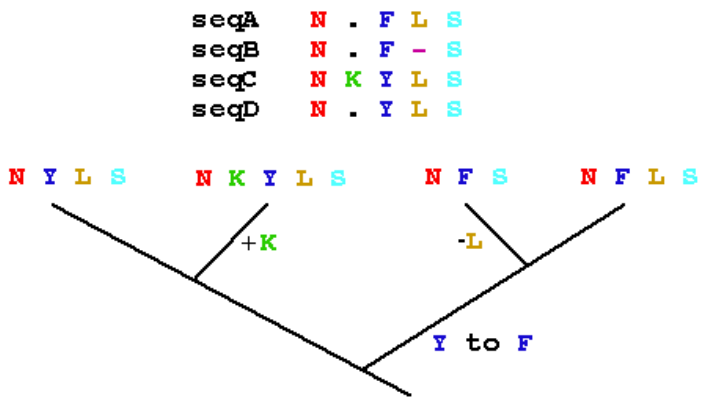
The directed lines connect the most probable sources and target localities of viral lineages



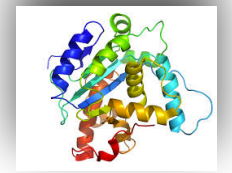
Alignment of E protein showing predicted glycosylation sites.

Multiple Sequence Alignment

- MSA can be used to start phylogenetic analysis. The phylogenetic tree describes the distance between the sequences under analysis. From the alignment, each column shows if there are conservation of the residues (amino-acids), mutations or divergence from the common ancestor.

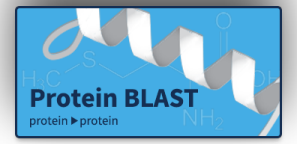


- Taken from http://genome.crg.es/courses/msa/

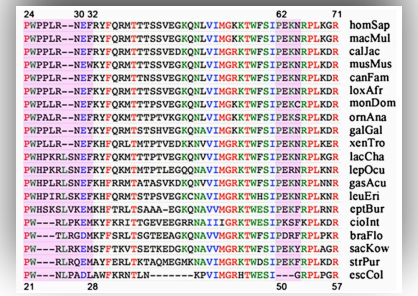


PWPPLRSTVVNEFWGH ...

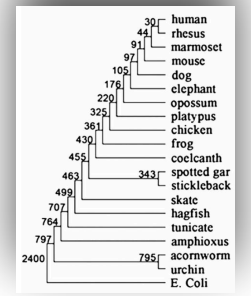
Protein of unknown function



BLAST



Multiple Sequence Alignment



Phylogenetic tree

- Functional significance of evolving protein sequence in dihydrofolate reductase from bacteria to humans
- Liu et al PNAS June 18, 2013 110 (25) 10159-10164;

The Phylogeny Problem

- Assume we are given a set of sequences evolutionarily related, i.e. with a common ancestor.
- The phylogeny problem: infer the best possible evolutionary tree.
- This is an optimisation problem since the number of possible trees increases exponentially with the number of input sequences.
- As optimisation problem it requires an objective function.

Classes of algorithms

- The three main classes of algorithms essentially differ in two aspects: way the objective function is calculated and the mechanism to search through the solution space.
- **Distance-based algorithm**: calculate a distance matrix based on the pairwise distances of the sequences. Derive trees based consistent with distances from the matrix.
- **Maximum parsimony**: search for trees that try to minimize the number of mutations (in internal nodes of the tree) to explain the variability of the sequences. Based on MSA of the input sequences. Use certain columns in the alignment that are informative of the possible phylogeny.
- **Statistical/Bayesian**: probabilistic models for the occurrence of different types of mutations in the sequences. Score trees based on their probability searching the most likely trees that explain the sequences according to the assumed model.

Distance-based methods

- Objective functions for this class of methods rely on measuring the consistency of the distances between the leaves in the tree (sequences) and the distances derived from sequence similarity (alignment).
- The structure of the tree and the length of the branches connecting the nodes reflect the pairwise distances between sequences.

- Step 1: *Calculate matrix of distances between sequences*.
  - Distance is the reverse of similarity (e.g. percentage of columns in the alignment with mismatches or gaps).
  - Objective function or error function: tries to minimize the difference between the distances in the tree and the distances in the matrix.

Objective function

$$score(T) = \sum_{i,j \in S} (d_{ij}(T) - D_{ij})^2$$

- S: set of input sequences
- T: tree
- $d_{ij}(T)$: distances of the leaves representing sequences i and j in the the tree.
- D_{ij}: distance between sequences I and j in the input matrix D given from sequence alignment.

- In a rooted phylogenetic tree, distances between nodes u and v is given by the distances traveled from u to v.
- If w is the nearest common ancestor than $d_{uv} = d_{uw} + d_{vw}$
- Distances are computed as the difference on the height of the nodes: $d_{uw} = h(w) - h(u)$,

where $h(x)$ denotes the height of the node x.

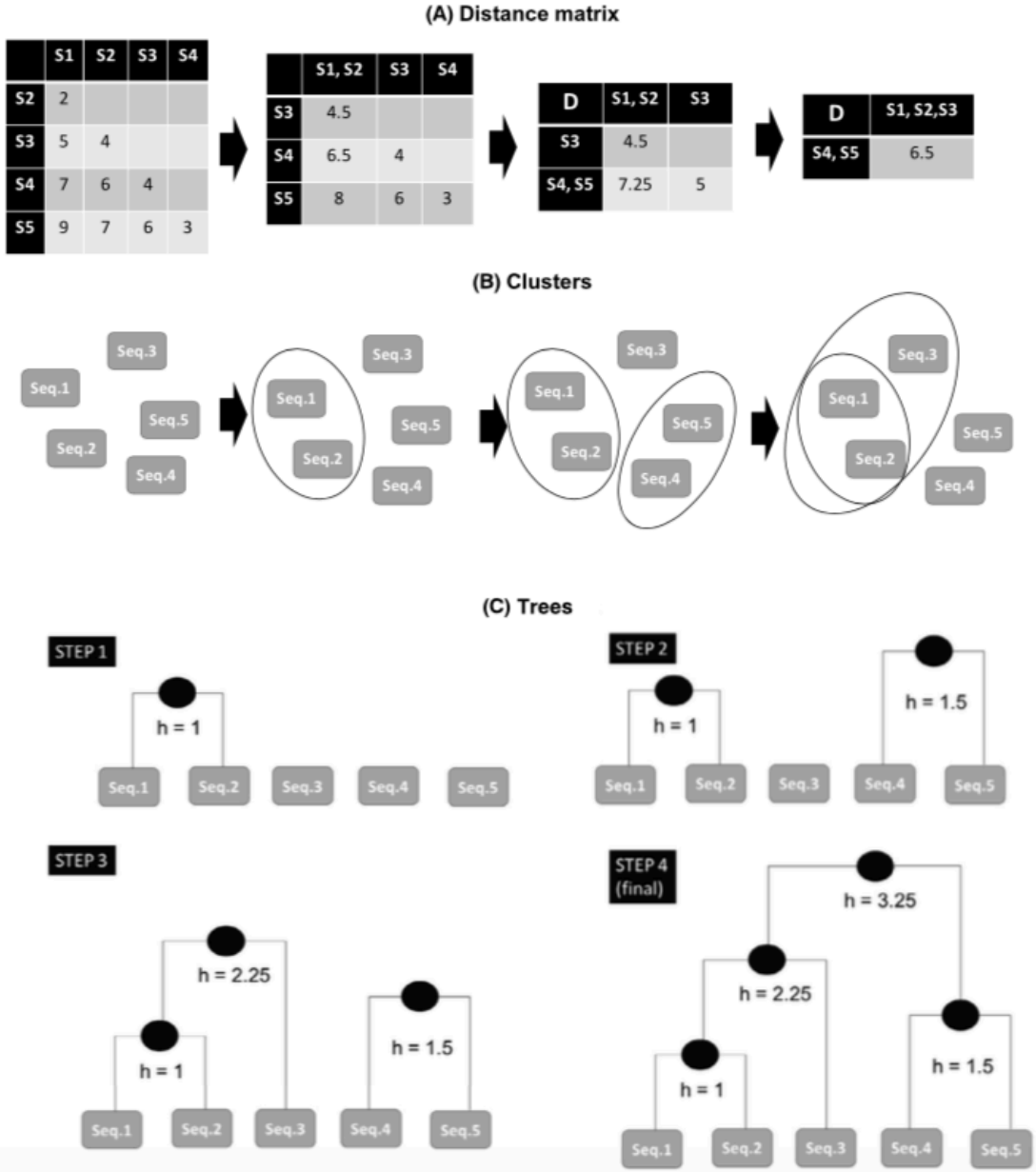
- $d(w)$ is larger than $d(v)$ and $d(u)$ since w is a common ancestor of u and v.
- Under the molecular clock hypothesis, the mutation rate in all branches of the tree is uniform.
- **Ultrametric tree**: the distance between all leaves and the root is the same. The height of each leaf is 0. Thus, $d_{uw} = 2 X h(w)$.

Methods

- As the number of sequences increases the solution space also increases exponentially. Heuristic methods need to be applied to obtain solutions in reasonable time.
- We will study the *Unweighted Pair Group Method Using Arithmetic Averages* (UPGMA), which is based on agglomerative hierarchical clustering algorithms.

- Clustering algorithm:
  - Consider each sequence (tree leaf) as its own cluster. height = 0 in the tree.
  - Merge the pair of closest sequence/clusters (minimum value in the matrix D); join these sequences creating an internal node. The height = half of distance between sequences. These sequences form a cluster.
  - Distance of a cluster to the remaining sequences is the average of the distances. Update distance matrix D: remove *cols* and *rows* of the connected sequences. Add *row* and *col* for the new cluster.
  - Iteratively: find pairs of clusters with minimum distance and repeat: join clusters, add internal node to the tree with the given height and update D.
  - Stop when all sequences are within a single cluster that corresponds to the node of the tree.

UPGMA



- From Bioinformatics Algorithms, Rocha & Ferreira

Distance between clusters

- In UPGMA the distance between clusters A and B:

$$\frac{1}{|A|.|B|}\sum_{i\in A}\sum_{j\in B} D_{ij}$$

  - From Bioinformatics Algorithms, Rocha & Ferreira

- If in a iteration clusters A and B are merged as A + B, the distance to any other cluster X can be given by the weighted average distance already calculated in the matrix:

$$D(A\cup B, X) = \frac{|A|.D(A,X) + |B|.D(B,X)}{|A| + |B|}$$

  - From Bioinformatics Algorithms, Rocha & Ferreira

- The WPGMA (Weighted Pair Group Method with Arithmetic Mean). The distances of new clusters to existing ones are calculated as the arithmetic mean of the distances of the joined clusters:

$$D(A\cup B, X) = \frac{D(A,X) + D(B,X)}{2}$$

  - From Bioinformatics Algorithms, Rocha & Ferreira

Exercises

- Consider the following multiple sequence alignment from a set of four sequences:

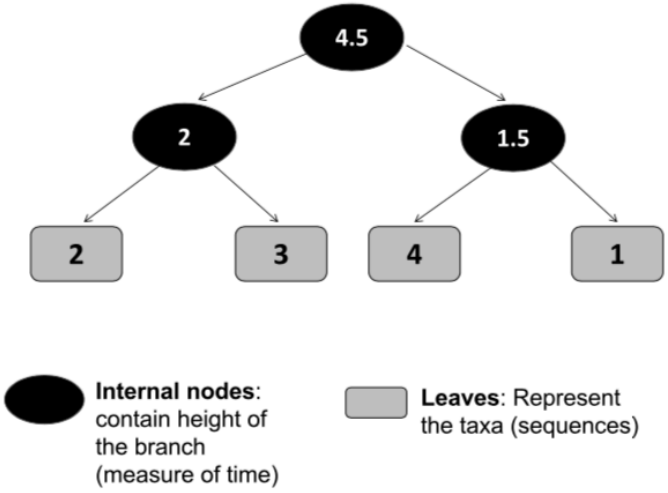
```
S1: A-CATATC-AT-
S2: A-GATATT-AG-
S3: AACAGATC-T--
S4: G-CAT--CGATT
```

- Calculate the distance matrix. Assume the metric distance as the number of distinct characters in pairwise alignment (assume the pairwise alignment given by the MSA).
- Build the tree for the sequences using the UPGMA algorithm.

Binary Trees

- Evolutionary trees will be represented as binary trees.
- BTs:
  - Recursive data structures
  - Every tree has a node (with some information) a left and right tree.
  - Each sub-tree null or represent another tree.
  - Leaves contain information on the nodes but both sub-trees are null.

- In phylogenetic trees:
  - Sequences are represented in the leaves.
  - Internal nodes represent mutation events.



- From Bioinformatics Algorithms, Rocha & Ferreira

Binary Trees

- Implement a Binary Tree Class
- Attributes:
  - Value - integer to keep the index of the sequence (not the sequence itself) to represent in the leaves; -1 for internal nodes.
  - Distance - height of the node (0 for the leaves).
  - left and right - left and right sub-trees; for leaves will be *None*.

```
1. class BinaryTree:
2.
3.     def __init__(self, val, dist = 0, left = None, right = None):
4.         self.value = val
5.         self.distance = dist
6.         self.left = left
7.         self.right = right
8.
```

Binary Trees

- Prints tree

```
1. def print_tree(self):
2.     self.print_tree_rec(0, "Root")
3.
4. def print_tree_rec (self, level, side):
5.     tabs = ""
6.     for i in range(level): tabs += "\t"
7.     if self.value >= 0:
8.         print(tabs, side, " - value:", self.value)
9.     else:
10.        print(tabs, side, "- Dist.: ", self.distance)
11.        if (self.left != None):
12.            self.left.print_tree_rec(level+1, "Left")
13.        if (self.right != None):
14.            self.right.print_tree_rec(level+1, "Right")
```

Binary Trees

- Write a function called **get_cluster** that given a tree returns all the elements (leaves) in the tree as list. Hint: Traverse tree and collect the elements in the leaves. If it is in an internal node (value == -1) call the function recursively.

Binary Trees

- Write a function called **get_cluster** that given a tree returns all the elements in the tree as list. Hint: Traverse tree and collect the elements in the leaves. If it is in an internal node (value == -1) call the function recursively.

```
1. def get_cluster(self):
2.     res = []
3.     if self.value >= 0:
4.         res.append(self.value)
5.     else:
6.         if (self.left != None):
7.             res.extend(self.left.get_cluster())
8.         if (self.right != None):
9.             res.extend(self.right.get_cluster())
10.    return res
11.
12.
```

list1.extend(list2): the elements of list2 are added to the end of list1.

Binary Trees


```
1. def size(self):
2.     '''size of the tree: returns two values
3.     - number of internal nodes of the tree
4.     - number of leaves'''
5.     numleaves = 0
6.     numnodes = 0
7.     if self.value >= 0:
8.         numleaves = 1
9.     else:
10.        if (self.left != None):
11.            resl = self.left.size()
12.        else: resl = (0,0)
13.        if (self.right != None):
14.            resr = self.right.size()
15.        else: resr = (0,0)
16.        numnodes += (resl[0] + resr[0] + 1)
17.        numleaves += (resl[1] + resr[1])
18.    return numnodes, numleaves
19.
```

Binary Trees

- Search the tree if exists a value. This value is passed as input. The output is a boolean indicating if exists or not. Implement method: **exists_leaf**

Binary Trees

- Search the tree if exists a value. This value is passed as input. The output is a boolean indicating if exists or not. Implement method: **exists_leaf**

```
1. def exists_leaf(self, leafnum):
2.     if self.value >= 0:
3.         if self.value == leafnum:
4.             return True
5.         else: return False
6.     else:
7.         if self.left != None:
8.             resl = self.left.exists_leaf(leafnum)
9.             if resl == True: return True
10.        if self.right != None:
11.            resr = self.right.exists_leaf(leafnum)
12.            if resr == True: return True
13.    return False
14.
```

Exercises

- Implement the method **common_ancestor** to calculate the simplest tree (with the less height) that contains the two input leaves. The input is two sequence identifiers. The output is a tree.
- Implement a method, **distance_leaves,** that calculates the distance between two leaves. Input: leaf1 and leaf2 ids. Returns an integer. Hint: use the common ancestor.

Exercises

- Implement the method **common_ancestor** to calculate the simplest tree (with the less height) that contains the two input leaves. The input is two sequence identifiers. The output is a tree.
- Implement a method, **distance_leaves,** that calculates the distance between two leaves. Input: leaf1 and leaf2 ids. Returns an integer. Hint: use the common ancestor.

```
1. def common_ancestor(self, leaf1, leaf2):
2.     if self.value >= 0: return None
3.     if self.left.exists_leaf(leaf1):
4.         if self.left.exists_leaf(leaf2):
5.             return self.left.common_ancestor(leaf1, leaf2)
6.         if self.right.exists_leaf(leaf2):
7.             return self
8.         return None
9.     if self.right.exists_leaf(leaf1):
10.        if self.right.exists_leaf(leaf2):
11.            return self.right.common_ancestor(leaf1, leaf2)
12.        if self.left.exists_leaf(leaf2):
13.            return self
14.    return None
15.
16.def distance_leaves(self, leafnum1, leafnum2):
17.    ca = self.common_ancestor(leafnum1, leafnum2)
18.    return 2*ca.distance
19.
```

NumMatrix

- NumMatrix is a class to keep and manipulate numerical matrices.
- Matrices are triangular (only cells with row index > col index contain information, the rest of the values are filled with zero).
- Methods to access the information including:
  - Number of rows/cols
  - Accessing and setting values given rows and cols
  - Print the matrix
  - Add or remove rows/cols
  - Provide a matrix copy
  - **min_dist_indexes**: row and col for the minimum value.

Hierarchical Clustering

- Implements a general purpose agglomerative hierarchical clustering. Distance matrix is an attribute.
- **execute_clustering** runs the algorithm and returns a binary tree.
  - Start:
    - Initialise the set of trees, creating the leaf nodes.
    - Calculate the distance matrix
  - Identify clusters to join:
    - Find indices of the minimum distance in the matrix
    - Create a new tree to join the clusters with minimum distance
  - If last iteration:
    - Return the tree
  - else:
    1. Remove from the list of trees the joined branches
    2. Update distance matrix removing cols and rows of the joined clusters; add a new one for the new cluster
    3. The new tree is added to the set of trees to handle in posterior iterations.

Hierarchical Clustering


```
1. from BinaryTree import BinaryTree
2. from NumMatrix import NumMatrix
3.
4. class HierarchicalClustering:
5.
6.     def __init__(self, matdists):
7.         self.matdists = matdists
8.
```

- Fill in the code in the class HierarchicalClustering.py

Hierarchical Clustering


```
1.  def execute_clustering(self):
2.      trees = []
3.      tableDist = self.matdists.copy()
4.      for i in range(self.matdists.num_rows()):
5.          t = BinaryTree(i)
6.          trees.append(t)
7.      for k in range(self.matdists.num_rows(), 1, -1):
8.          mins = tableDist.min_dist_indexes()
9.          i,j = mins[0], mins[1]
10.         n = BinaryTree(-1, tableDist.get_value(i, j)/2.0, trees[i], trees[j])
11.         if k>2:
12.             ti = trees.pop(i)
13.             tj = trees.pop(j)
14.             dists = []
15.             for x in range(tableDist.num_rows()):
16.                 if x != i and x != j:
17.                     si = len(ti.get_cluster())
18.                     sj = len(tj.get_cluster())
19.                     d = (si*tableDist.get_value(i,x) + sj*tableDist.get_value(j,x)) / (si+sj)
20.                     dists.append(d)
21.             tableDist.remove_row(i)
22.             tableDist.remove_row(j)
23.             tableDist.remove_col(i)
24.             tableDist.remove_col(j)
25.             tableDist.add_row(dists)
26.             tableDist.add_col([0] * (len(dists)+1))
27.             trees.append(n)
28.         else: return n
```

UPGMA

- Applies the generic hierarchical clustering algorithm to the given biological sequences.
  - Attributes:
    - Sequences to analyse (leaves of the tree) - MySeq objects
    - Alignment parameters - object PairwiseAlignment
    - Distance matrix - object NumMatrix

```
1. from NumMatrix import NumMatrix
2. from HierarchicalClustering import HierarchicalClustering
3. from MySeq import MySeq
4. from PairwiseAlignment import PairwiseAlignment
5. from SubstMatrix import SubstMatrix
6.
7. class UPGMA:
8.
9.     def __init__(self, seqs, alseq):
10.        self.seqs = seqs
11.        self.alseq = alseq
12.        self.create_mat_dist()
13.
14.
```

- Complete the **create_mat_dist** function.

UPGMA


```
1. def create_mat_dist(self):
2.     self.matdist = NumMatrix(len(self.seqs), len(self.seqs))
3.     for i in range(len(self.seqs)):
4.         for j in range(i, len(self.seqs)):
5.             s1 = self.seqs[i]
6.             s2 = self.seqs[j]
7.             self.alseq.needleman_Wunsch(s1, s2)
8.             alin = self.alseq.recover_align()
9.             ncd = 0
10.            for k in range(len(alin)):
11.                col = alin.column(k)
12.                if (col[0] != col[1]): ncd += 1
13.            self.matdist.set_value(i, j, ncd)
14.
15.def run(self):
16.    ch = HierarchicalClustering(self.matdist)
17.    t = ch.execute_clustering()
18.    return t
19.
```

Exercises

- Consider the following sequences used in the previous exercise. Write python code to verify your previous results.

ACATATCAT

AACAGATCT

AGATATTAG

GCATCGATT