Algorithm for Construction of Phylogenetic Tree Using BioPython

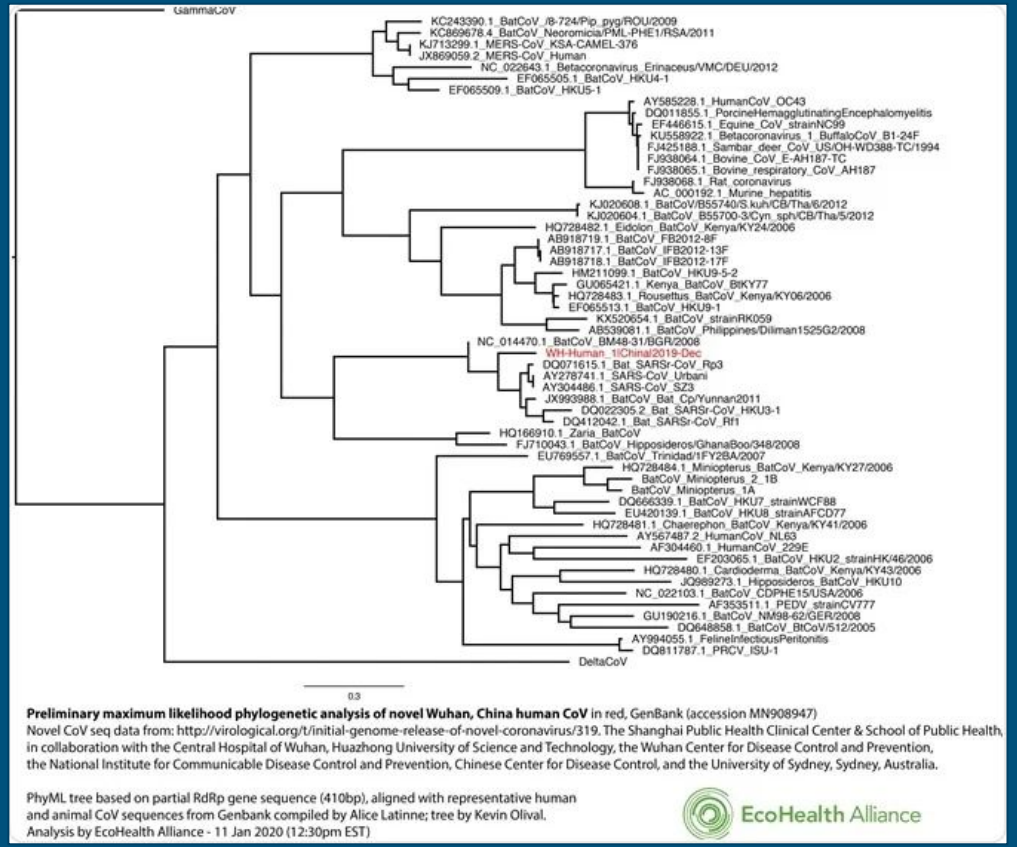
Liam Benjamin
N'yoma Diamond
Emmaline Raven
Charlotte Clark

11 December 2020

Introduction

Phylogenetic Trees

- Spatially represent evolutionary relationships
- Built from sequences (protein, DNA, etc.)
- Long sequences can be unwieldy
- How can we make this process as convenient as possible?



Phylogenetic analysis of COVID-19, in red.
From https://www.news-medical.net/health/The-Phylogenetic-Tree-of-the-SARS-CoV-2-Virus.aspx

Methodology

BioPython Library

- A variety of computational biology functions
- Most useful modules for us:
  - ClustalW and AlignIO - for MSA
  - Phylo - for working with phylogenetic trees, uses common modules like Matplotlib to **draw** phylogenetic trees for visualization
  - Pandas - for data manipulation

```
from Bio import AlignIO
from Bio import Phylo
from Bio.Align import substitution_matrices
from Bio.Align.Applications import ClustalwCommandline
from io import StringIO
from numpy import NaN
import pandas as pd
import re
```

Sequence Alignment

```
# create a multiple sequence alignment from provided file
def msa(filePath):
    ClustalwCommandline("clustalw2", infile=filePath, clustering="UPGMA")()
    return AlignIO.read(filePath.split(".")[0] + ".aln", "clustal")
```

- ClustalW will accept a variety of file types, including FASTA
- `msa(filePath)` returns a multiple sequence alignment in a special Sequence Alignment type that you can iterate through (similar to a list).
    - This type stores characteristics of each sequence, such as name, description, etc.

FASTA Input:

>Diplocarpon_rosae
MPPKSAPKAKAASSNTTPGPSYQDMIIHAITTLKDRKGSSRIVLKKFVKSNNDIKAGD
SMFTSLFNKAIAVGVEKGVFEQPKGASGTVKLAQKAPKPAAAKAAAKPKVEKKTTEK
KPAVKKAPVAKKTTATKVVKPKADPTPKASPKAKAAAKPKAASKAKVAQPKVAAKP
KKASAAKAVPAVVEKPSVLNKTKSGRVTKTTSATPKKAAAKKATPKKKATPKKA
>Oncorhynchus_mykiss
MAEVAPAPAAAAPAKAPKKKAAAKPKKAGPSVGELIVKAVSASKERSGVSLAALKKS
LAAGGYDVEKNNSRVKIAVKSLVTKGTLVQTKGTGASGSFKLNKKAVEAKKPAKKAA
APKAKKVAAKKPAAAKKPKKVAAKKAVAAKKSPKKAKKPATPKKAAKSPKKVKKPA
AAAKKAAKSPKKATKAAKPKAAKPKAAKAKKAAPKKK
. . .

Converting BLOSUM62 to Distance

```
def getBLOSUMDistanceMatrix(alignment):
    blosumMatrix = substitution_matrices.load("BLOSUM62")

    df = pd.DataFrame(columns=list(r.id for r in alignment),index=list(r.id for r in alignment))

    for record1 in alignment:
        for record2 in alignment:
            score = 0
            for i in range(len(record1.seq)):
                aa1 = record1[i] if record1[i] != '-' else '*'
                aa2 = record2[i] if record2[i] != '-' else '*'
                score -= blosumMatrix[aa1][aa2] #subtracts so closer nodes have lower values
            df[record1.id][record2.id] = score

    return df.apply(pd.to_numeric)
```

- Align subpackage - substitution_matrices includes most useful matrices (BLOSUM variations, PAM variations, etc)
- Give alignment generated from `msa( )`, returns a distance matrix ready for UPGMA
- Uses *negative* BLOSUM scores because lower BLOSUM scores are “worse” (farther)

About Distance Matrices

	a	b	c	d	e
a	0	17	21	31	23
b	17	0	30	34	21
c	21	30	0	28	39
d	31	34	28	0	43
e	23	21	39	43	0

	(a,b)	c	d	e
(a,b)	0	25.5	32.5	22
c	25.5	0	28	39
d	32.5	28	0	43
e	22	39	43	0

"Cluster" - Column/row. Represents a species or strain

Cluster distance - Value at intersection

Merging - Average distance to all sub-clusters

Example matrices sourced from https://en.wikipedia.org/wiki/UPGMA

UPGMA

Given a distance matrix:

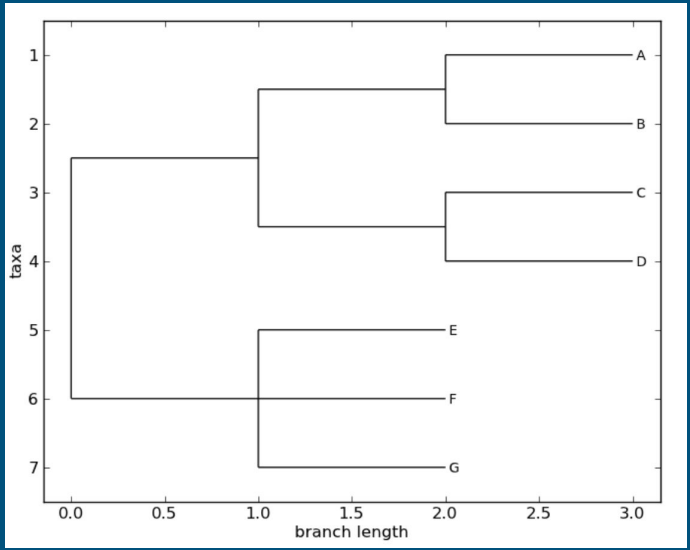
1. While the distance matrix has more than 1 cluster:
    - Ci, Cj <- closest clusters
    - Cnew <- Merge Ci and Cj
    - Create node representing Cnew (Ci and Cj are children of new node)
    - Set the age of the new node to distance between closest clusters
    - Add Cnew to distance matrix
    - Remove Ci and Cj from the distance matrix
2. Traverse graph starting at the remaining cluster in the distance matrix
    - Distance between any two nodes is the difference in their age

Visualizing the Tree

Newick format - representing trees using parentheses and commas, recognized by BioPython

```
(((A,B),(C,D)),(E,F,G));
```

Looks like:



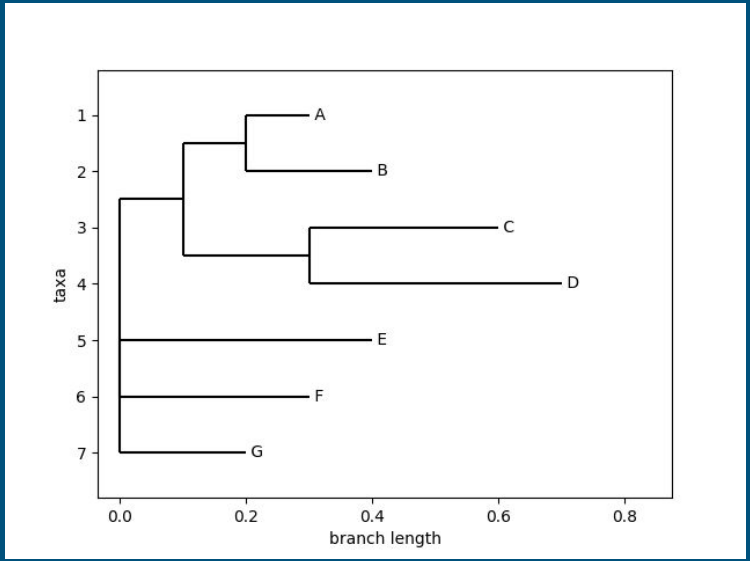
Branch lengths are assumed to be 1.0 when none are specified

Parentheses denote clusters.

Visualizing the Tree

To denote distance of each node from parent:

```
(((A:0.1,B:0.2):0.1,(C:0.3,D:0.4):0.2):0.1,(E:0.4,F:0.3,G:0.2));
```



*Note: there are some more robust formats (e.g. phyloxml), but Newick is simple and easy to work with in strings.

Phylo in biopython assumes 0 distance if unspecified when distances are specified elsewhere

So, we need to build a tree using the **unweighted pair group method with arithmetic mean (UPGMA)** method, then draw it.

Visualizing the Tree

```
s = nodes[distances.columns[0]].toString()
tree = Phylo.read(StringIO(s), "newick")
Phylo.draw(tree)
```

- Phylo uses common libraries (like Matplotlib) to visualize trees
  - *Note: you can also visualize the tree in ASCII in command line

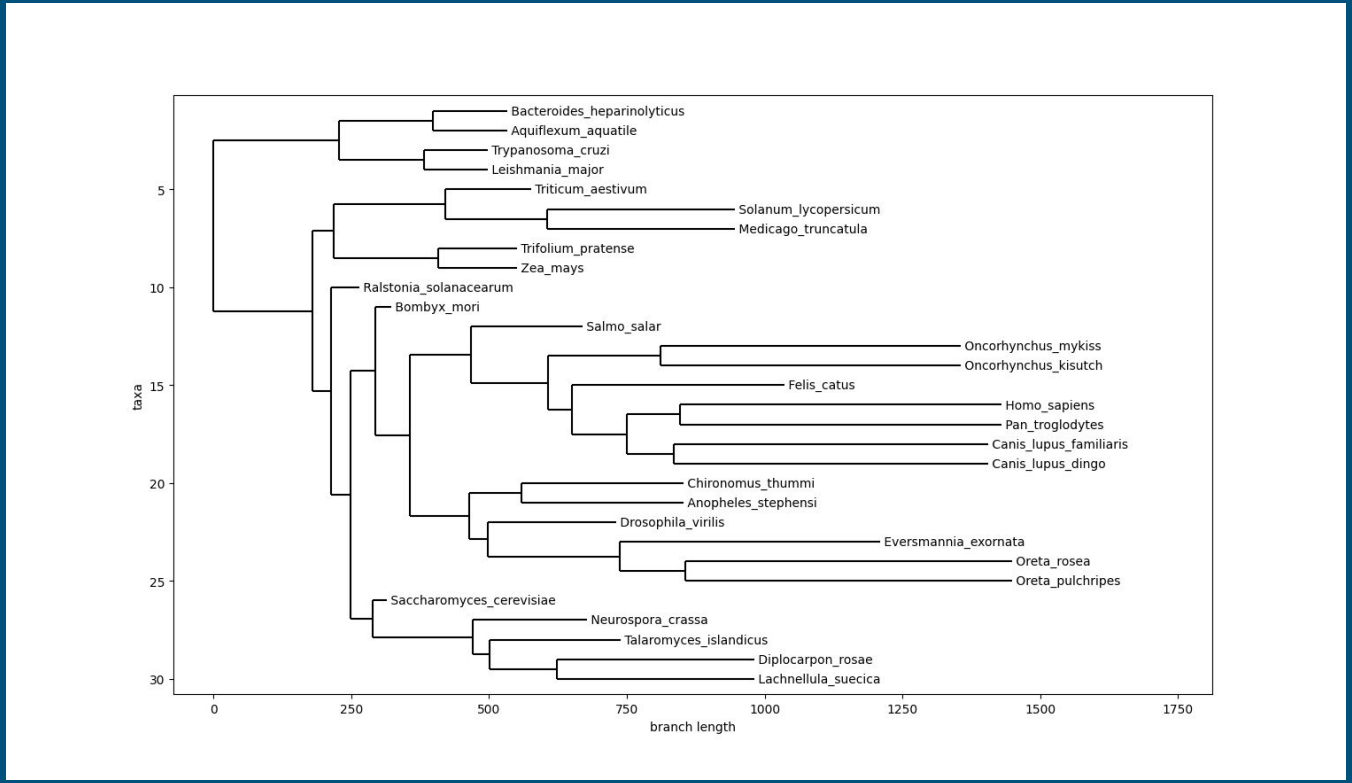
Implementation

```
def UPGMA(distances):
    #initialization
    labels = distances.columns

    for cluster in distances.columns:
        distances[cluster][cluster] = NaN

    nodes = {}

    for name in labels:
        nodes[name] = UPGMANode(name)

    Cnew = len(distances)
    #end initialization
```

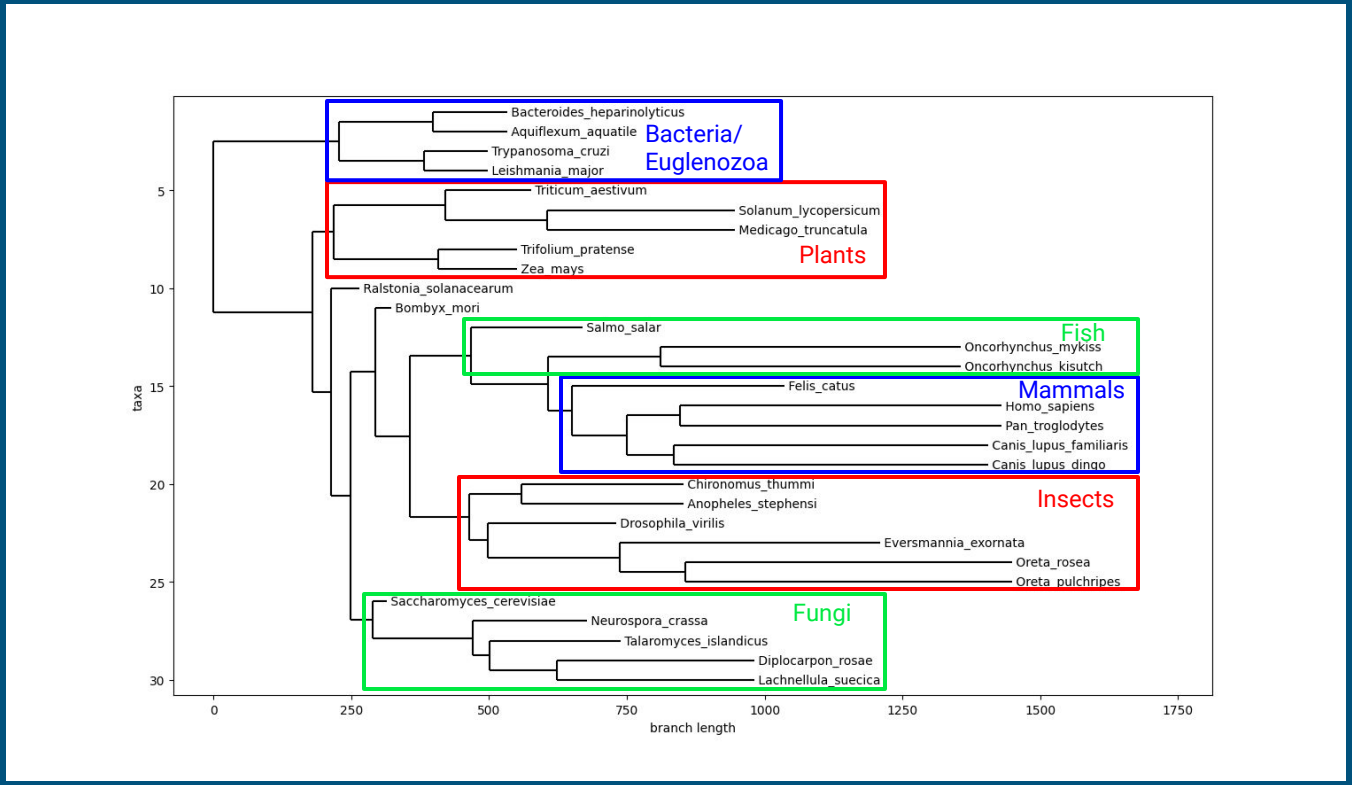
```
    while distances.size > 1:
        Ci, Cj = distances.stack().idxmin()

        distances[Cnew] = mergeClusters(Ci, Cj)
        distances.loc[Cnew] = distances[Cnew]

        nodes[Cnew] = UPGMANode(Cnew)

        nodes[Cnew].children.add(nodes[Ci], nodes[Cj])
        nodes[Ci].parent = nodes[Cnew]
        nodes[Cj].parent = nodes[Cnew]

        nodes[Cnew].age = distances[Ci][Cj]

        distances.drop(Ci, Cj)

        Cnew += 1

    s = nodes[distances.columns[0]].toString()
    tree = Phylo.read(StringIO(s), "newick")
    Phylo.draw(tree)
```

Results

Histone H1 Tree using BLOSUM Distances

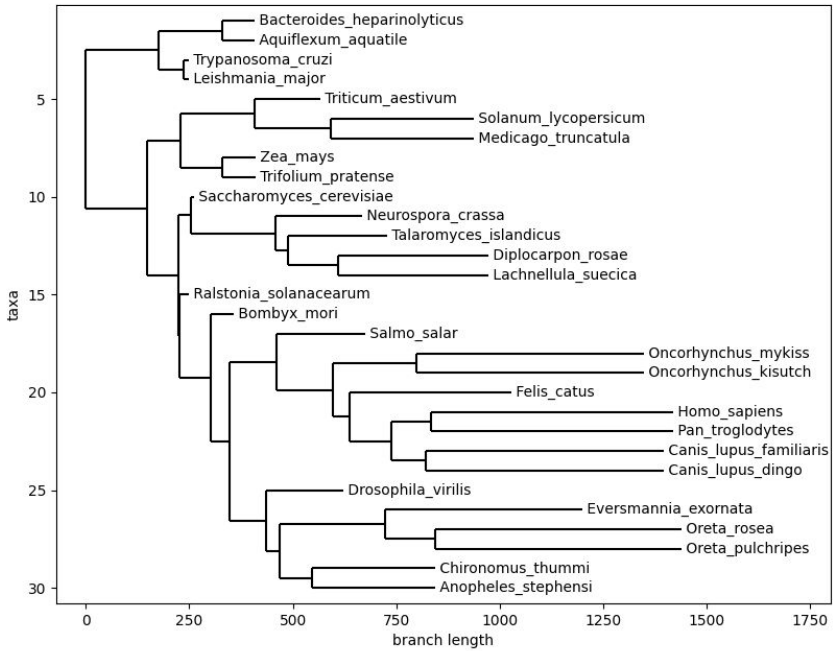


Histone H1 Tree using BLOSUM Distances

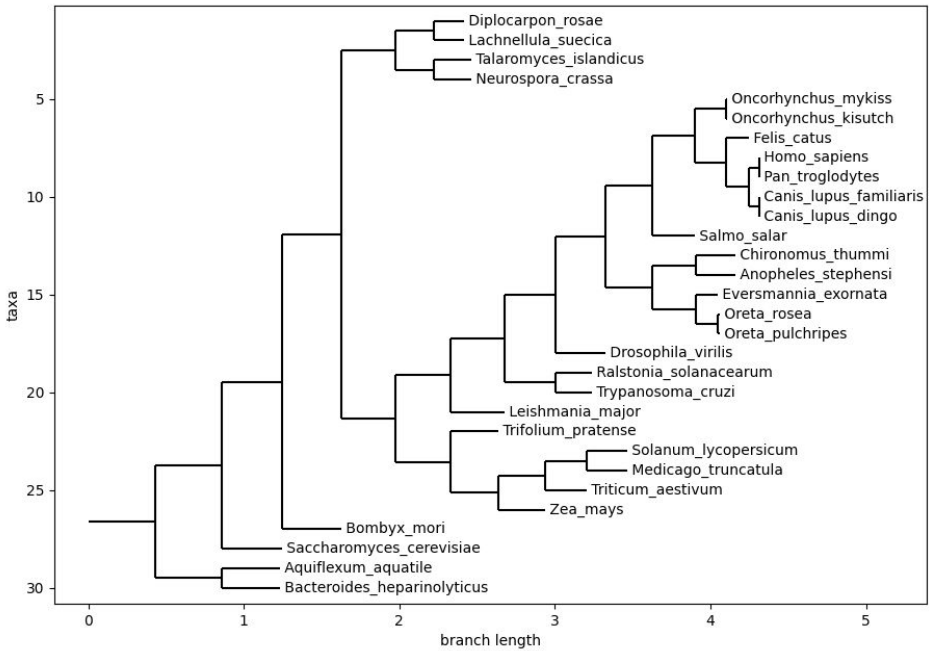


Compared to ClustalW UPGMA

Ours



ClustalW



Conclusions

Conclusions

- BioPython methods and tools greatly simplify computational biology applications
  - AlignIO for MSA
  - Phylo for building and drawing trees
  - UPGMA is NOT in BioPython - good example of BioPython’s strengths
- UPGMA is theoretically simple but is tedious
  - A working Python program greatly reduces this tedium

References

https://www.bioinformaticsalgorithms.org/

https://biopython.org/wiki/Documentation

https://github.com/biopython/biopython

https://biopython.org/docs/1.74/api/

https://en.wikipedia.org/wiki/UPGMA

http://rosalind.info/problems/ba7d/

https://www.news-medical.net/health/The-Phylogenetic-Tree-of-the-SARS-CoV-2-Virus.aspx