CS 5 Green Today

HMC CS 5 Green Professors discover new discovery

Claremont, CA: Researchers at Harvey Mudd College have made an extremely important new discovery said a spokesperson for the College. The discovery was evidently discovered while the researchers were trying to discover another discovery. “The professors discovered that their discovery had not been previously discovered, which is an important discovery in its own right,” said the excited spokesperson. A number of prominent scientists also expressed their tremendous enthusiasm and said that they looked forward to reading what was actually discovered.
Learning Goals

• Review functions on trees
• Introduce a distance-based approach to phylogenetic tree reconstruction (UPGMA)
Phylogenetic Trees...

From Darwin’s notebooks, 1837
I think

D1

B C

E
Really?
The only figure in *On the Origin of Species by Natural Selection* (Darwin, 1859)

From the 6th edition (1872): “The affinities of all the beings of the same class have sometimes been represented by a great tree. I believe this simile largely speaks the truth. The green and budding twigs may represent existing species; and those produced during former years may represent the long succession of extinct species.”
More recursion on trees: mrca

```python
>>> find("L", groodies)
True

>>> mrca("L", "E", groodies) # use find as a helper!
'Z'

>>> mrca("W", "E", groodies)
'Y'

>>> mrca("W", "C", groodies)
'X'

>>> mrca("W", "Prof Wu", groodies)
None
```
def mrca(species1, species2, tree):
    """Return the name of the most recent common ancestor of species1 and species2. If there isn't one, return None."""
    root, left, right = tree
You should use `find` here...

```python
def mrca(species1, species2, tree):
    """Return the name of the most recent common ancestor of species1 and species2. If there isn't one, return None.""
    root, left, right = tree
    if left == ():
        return None
    elif (not find(species1, tree)) or (not find(species2, tree)):
        return None
    else:
        if find(species1, left) and find(species2, left):
            return mrca(species1, species2, left)
        elif find(species1, right) and find(species2, right):
            return mrca(species1, species2, right)
        else:
            return root
```
**Input:** DNA or protein sequences for each species

**Output:** Species tree

Distance-Based Approach

Groody  CATCAACCAGTGAACCAGTATAGGACGCC
Froody  CAACACTCAGTGACAAGTCTAGCACGCC
Snoody  AATCGCCCGGCGTCAGGCATAGCTAGCGC

\[
\begin{array}{ccc}
G & F & S \\
G & 0 & 6 & 12 \\
F & 6 & 0 & 12 \\
S & 12 & 12 & 0 \\
\end{array}
\]
Distance-Based Approach

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>F</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>S</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

Current time (“time 0”)

(6.0,
 ('Snoody', (), ()),
 (3.0,
    ('Groody', (), ()),
    ('Froody', (), ()))
)

(6.0,
 ('Snoody', (), ()),
 (3.0,
    ('Groody', (), ()),
    ('Froody', (), ()))
)

Snoody

Groody

Froody
### Unweighted Pair Group Method with Arithmetic Mean (UPGMA) Algorithm

Groody  
Froody  
Snoody  

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>F</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>S</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>
Unweighted Pair Group Method with Arithmetic Mean (UPGMA) Algorithm

Groody
Froody
Snoody

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>F</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>S</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
D_1((G,F),S) = \frac{D_0(G,S) + D_0(F,S)}{2}
= \frac{12 + 12}{2} = 12
\]
Try This One…

Aoody
Boody
Coody
Doody
Eoody

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

This algorithm’s “bark” is worse than it’s bite.

The matrix is symmetric, so we just need to keep the bottom or top half!
Try This One…

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>16</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(A,B)</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A,B)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>16</td>
<td>8</td>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(A,B)</th>
<th>C</th>
<th>(D,E)</th>
<th>D3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(A,B)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(D,E)</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(A,B)</th>
<th>(C, (D,E))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A,B)</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(C, (D,E))</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>
DEMO!

I wood love to see this in action!
Implementing UPGMA

Let’s see this in the provided mitoData.py file

N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0,  (N1, N2):6,    (N1, N3):12,
(N2, N1):6,  (N2, N2):0,    (N2, N3):12,
(N3, N1):12, (N3, N2):12,  (N3, N3):0}
1. For all *pairs* of trees in the `groodies_lst`, find the closest pair.
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0, (N1, N2):6, (N1, N3):12,
(N2, N1):6,  (N2, N2):0, (N2, N3):12,
(N3, N1):12, (N3, N2):12, (N3, N3):0}

1. For all *pairs* of trees in the groodies_lst, find the closest pair

   N1 = ("Groody", (), ())
   N2 = ("Froody", (), ()
N1 = ("Groody", (), ()), N2 = ("Froody", (), ()), N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0, (N1, N2):6, (N1, N3):12,
(N2, N1):6, (N2, N2):0, (N2, N3):12,
(N3, N1):12, (N3, N2):12, (N3, N3):0}

1. For all pairs of trees in the groodies_lst, find the closest pair

   N1 = ("Groody", (), ()), N2 = ("Froody", (), ()), N3 = ("Snoody", (), ())

groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0, (N1, N2):6, (N1, N3):12,
(N2, N1):6, (N2, N2):0, (N2, N3):12,
(N3, N1):12, (N3, N2):12, (N3, N3):0}

2. Remove those from the groodies_lst

   groody_lst.remove(N1)
groody_lst.remove(N2)
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1): 0,  (N1, N2): 6,  (N1, N3): 12,
               (N2, N1): 6,  (N2, N2): 0,  (N2, N3): 12,
               (N3, N1): 12, (N3, N2): 12, (N3, N3): 0}

1. For all pairs of trees in the groodies_lst, find the closest pair

   N1 = ("Groody", (), ())
   N2 = ("Froody", (), ())

2. Remove those from the groodies_lst

3. Make a new tree by joining these two trees...
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N1, N2, N3]
groodies_mat = {(N1, N1):0, (N1, N2):6, (N1, N3):12, 
               (N2, N1):6, (N2, N2):0, (N2, N3):12, 
               (N3, N1):12, (N3, N2):12, (N3, N3):0}

1. For all *pairs* of trees in the *groodies_lst*, find the closest pair

   N1 = ("Groody", (), ())
   N2 = ("Froody", (), ())

2. Remove those from the *groodies_lst*

3. Make a new tree by joining these two trees...

   new_tree = (3.0, N1, N2)
               = (3.0, ("Groody", (), ()), ("Froody", (), ()))
1. For all pairs of trees in the `groodies_lst`, find the closest pair.

   N1 = ("Groody", (), ())
   N2 = ("Froody", (), ())

   `groodies_lst` = `[N3, (3.0, ("Groody", (), ()), ("Froody", (), ()))]

   `groodies_mat` = {
                   (N1, N1):0,  (N1, N2):6,  (N1, N3):12,
                   (N2, N1):6,  (N2, N2):0,  (N2, N3):12,
                  }

2. Remove those from the `groodies_lst`

3. Make a new tree by joining these two trees...

   `new_tree` = (3.0, N1, N2)
   = (3.0, ("Groody", (), ()), ("Froody", (), ()))

4. Add this new tree to the `groodies_lst`
N1 = ("Groody", (), ())
N2 = ("Froody", (), ())
N3 = ("Snoody", (), ())
groodies_lst = [N3, (3.0, ("Groody", (), ()), ("Froody", (), ()))]
groodies_mat = {
    (N1, N1):0, (N1, N2):6, (N1, N3):12,
    (N2, N1):6, (N2, N2):0, (N2, N3):12,
}

5. Update the distance matrix...

groodies_mat[(N3, new_tree)] = ...
groodies_mat[(new_tree, N1)] = ...

We are still holding on to N1 and N2, even though we have removed them from the groodies_lst!
Implementing UPGMA

\texttt{findClosestPair( speciesList, Distances )}:
\begin{quote}
""Takes a list of species trees and the distance dictionary as input and returns a tuple \((X, Y)\) where \(X\) and \(Y\) are in the list and have the minimum distance between any two items in the list.""
\end{quote}

\texttt{updateDist( speciesList, Distances, newTree)}:
\begin{quote}
""Takes a list of species trees, the distance dictionary, and a newTree that was just formed by merging two trees found by \texttt{findClosestPair}. Those two trees can be found by looking inside \texttt{newTree}. Those two trees are removed from the distance dictionary and the \texttt{newTree} is added to the dictionary.""
\end{quote}

\texttt{upgma( speciesList, Distances)}:
\begin{quote}
""Returns the phylogenetic tree constructed by the UPGMA algorithm.""
\end{quote}
Updating the distance dictionary: a subtlety

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>4</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>12</td>
<td>12</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a,b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a,b</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>12</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a,b,c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>a,b,c</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

3
2

---

c__a__b

---
Updating the distance dictionary: a subtlety

When we combine distances between nodes with different numbers of leaves, we should weight by the number of leaves.

\[
\text{T1} = (a, b) \\
\text{T2} = c \\
\text{T3} = d
\]

new_tree = ("Anc", T1, T2)

dist(new_tree, T3) = \text{dist}(T1, T3) \times \frac{\text{leaf_count}(T1)}{\text{leaf_count(newTree)}} + \text{dist}(T2, T3) \times \frac{\text{leaf_count(T2)}}{\text{leaf_count(newTree)}}
Inferring time...

The \textit{scale} function you write in lab will be useful...
Rhea
(South America)

Ostrich
(Africa)

https://en.wikipedia.org/wiki/Ostrich
https://en.wikipedia.org/wiki/Gondwana
UPGMA assumes a molecular clock

<table>
<thead>
<tr>
<th></th>
<th>G</th>
<th>F</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>0</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>F</td>
<td>6</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>S</td>
<td>12</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>
A note on biogeography/migrations