Outline

1. Review of lecture 16 & additivity
2. Large datasets: Heuristic searching
3. Improvements on standard heuristics

Review

1. Two data types: Distance & Character
2. Two ways to “get a tree”: Clustering methods & Optimality Criteria methods (building vs searching)
3. All clustering methods are distance based - But not all distance methods are clustering methods (Minimum Evolution)

Additivity

- UPGMA requires ultrametric data
- NJ & ME require additive data
- All ultrametric data are additive but not the reverse
- A complete record of all genetic events would constitute a set of perfectly additive distances
- Thus observed distances are only estimates of true distances

Four steps - each should be explained in methods

1. Character (data) selection (not too fast, not too slow) “Why did you choose these data?”
2. Alignment of Data (hypotheses of primary homology) “How did you align your data?”
3. Analysis selection (choose the best model / method(s)) - data exploration “Why did you choose your analysis method?”
4. Conduct analysis - strategies etc.

Review Distance methods

- UPGMA - pairwise clustering method that requires ultrametric data - most data are not ultrametric = violation of method’s assumptions
- Neighbor-Joining - pairwise clustering method that does not require ultrametric data - only a fast approximation of the Minimum Evolution tree
- Minimum Evolution - "parsimony for distance data" an optimality criterion method - tree space is searched for the optimal tree that minimizes the sum of the branch lengths

Additivity

- UPGMA requires ultrametric data
- NJ & ME require additive data
- All ultrametric data are additive but not the reverse
- A complete record of all genetic events would constitute a set of perfectly additive distances
- Thus observed distances are only estimates of true distances
If there were no sampling (random) errors nor multiple hits then data would be additive & NJ / ME would work perfectly

\[ d = d_{A} + d_{B} \]

Additive properties:
- \( d_{AB} = d_{1} + d_{2} \)
- \( d_{AC} = d_{1} + d_{3} + d_{4} \)
- \( d_{AD} = d_{1} + d_{3} + d_{5} + d_{6} \)
- \( d_{BC} = d_{2} + d_{3} + d_{4} \)
- \( d_{BD} = d_{2} + d_{3} + d_{5} + d_{6} \)
- \( d_{CD} = d_{4} + d_{5} + d_{6} \)

Ulmometric properties:
- \( d_{1} = d_{1} \)
- \( d_{1} + d_{3} = d_{1} + d_{3} \)
- \( d_{1} + d_{3} + d_{5} = d_{1} + d_{3} + d_{5} \)

If 11 or fewer OTUs can do an **exhaustive search**
- this guarantees the shortest tree(s) will be found (an exact solution)
- every possible tree for n taxa examined
- slowest and most rigorous method
- provides a frequency histogram of tree scores

**Exhaustive Search**

Step 1
- Starting tree, any 3 taxa
- Add fourth taxon (D) in each of three possible positions -> three trees

Step 2a
- Add fifth taxon (E) in each of the five possible positions on each of the three trees - 15 trees, and so on ...

At 100k trees/sec PAUP would take over 2 billion years to evaluate all trees for 21 OTUs

<table>
<thead>
<tr>
<th>Number of OTUs</th>
<th>Number of unrooted trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>105</td>
</tr>
<tr>
<td>7</td>
<td>960</td>
</tr>
<tr>
<td>8</td>
<td>16,286</td>
</tr>
<tr>
<td>9</td>
<td>129,153</td>
</tr>
<tr>
<td>10</td>
<td>2,857,602</td>
</tr>
<tr>
<td>11</td>
<td>34,459,425</td>
</tr>
<tr>
<td>12</td>
<td>534,729,075</td>
</tr>
<tr>
<td>13</td>
<td>13,748,345,573</td>
</tr>
<tr>
<td>14</td>
<td>316,224,145,225</td>
</tr>
<tr>
<td>15</td>
<td>7,385,635,589,625</td>
</tr>
<tr>
<td>16</td>
<td>212,458,046,876,875</td>
</tr>
<tr>
<td>17</td>
<td>4,182,282,391,628,375</td>
</tr>
<tr>
<td>18</td>
<td>91,804,705,562,274,625</td>
</tr>
<tr>
<td>19</td>
<td>2,332,606,879,732,950,625</td>
</tr>
<tr>
<td>20</td>
<td>52,142,290,476,699,771,875</td>
</tr>
<tr>
<td>21</td>
<td>1,286,794,512,637,661,359,375</td>
</tr>
</tbody>
</table>

For n taxa
- # trees = \( \frac{(2n-5)!}{(n-3)!} \)
- \( 2^{n(n-3)} \)

**Tree searching**

Finding the optimal trees belongs in a class of famously difficult problems called **NP-complete problems**

- These problems can't be solved with a simple equation - they are intractable for large N
- We are limited to the inelegance of “searching” for the solution - trying every possible solution
- A common example is the “travelling salesman problem”:
  - A salesman wants to visit N cities (each only once) and return home but wants to take the shortest route

If 11-25 OTUs can do a **branch and bound** search
- this also guarantees the shortest tree(s) will be found but not all trees are examined (also an exact solution)
- families of trees that cannot lead to shorter trees are discarded and not examined - saves time
- read text p. 174 (fig. 7.2) for details on method
- faster than exhaustive search
- no histogram of tree scores
Branch & Bound

At start, we know this tree has

Tree searching

For more than 25 OTUs (most datasets) must use other methods, **heuristic searching - approximate methods**
- do not guarantee the shortest tree will be found
  - but work quite well when they can be compared to exact methods
- fastest method (but less rigorous)
- many issues to consider to employ best strategy for searching tree space
- can get trapped in local optima while searching for global optima (shortest trees)

Heuristic Tree Searching

**heuristic searching - approximate methods**

1. A starting tree is obtained by some clustering method (or, rarely is totally random)
   eg stepwise addition or neighbor-joining

2. This tree is then subjected to **branch swapping** (movement of branches to new places on tree)
   - each swap makes a new topology which is scored using the Optimality Criterion
   - hope is to find a more optimal tree through extensive branch swapping

   **Somewhat analogous...**
   - The team member is like one "starting tree"
   - Climbing uphill = starting tree improved via branch swapping
   - Stopping at top of peak = starting tree has been "swapped to completion" and the best tree(s) found
   - No way to know if the peak is the highest one (fog too thick!)

Tree Islands

Imagine this depicts "tree space"

Heuristic search eigen types are "hill climbers" - they only climb up

search started here will not find global optimum
Heuristic Tree Searching
Hill climbing - Greedy Algorithms
- These algorithms are called “greedy” by computer scientists (but a better name would be “algorithms that lack foresight”)
- Because they choose whatever looks best at the moment and are unable to predict if that choice will be the best overall
- The best choice of a greedy algorithm may lead to suboptimal trees (trapped in a local optimum) for that search

Heuristic Tree Searching
Back to the analogy...
- An obvious solution, if you want to find the highest peak in the fog
- Is to drop lots of parachutists!
- And drop them in scattered locations across the landscape (rather than all in the same place)

Heuristic Tree Searching
Stepwise addition - builds starting tree (PAUP options)
(like deciding where to drop that parachutist)

Asis - the order in the data matrix (poor start unless you’ve sorted the OTUs in some phylogenetic order - which, of course, may bias your results!)
- produces same starting tree each time it is used

Closest -starts with shortest 3-taxon tree adds taxa in order that produces the least increase in tree length (greedy heuristic, like NJ) - will produce a ‘good’ starting tree but produces same starting tree each time it is used (unless there are ‘ties’ which are randomly broken)

Heuristic Tree Searching
Stepwise addition - builds starting tree (PAUP options)

Simple - the first taxon in the matrix is a taken as a reference - taxa are added to it in the order of their decreasing similarity to the reference
- again, produce same starting tree each time unless there are ties

Random - taxa are added in a random sequence, typically one would perform many replicates each starting with random addition of taxa - most rigorous
- each time a different starting tree can be produced
- like dropping lots of parachutists in different locations

I recommend random with at least 100 replicates (look for a plateau in results - same tree(s) found each time)

Heuristic Tree Searching
Multiple Random Stepwise Addition Sequence searches
- PAUP might report that there are multiple islands of same “height”

= the optimal trees on these are of the same score but valleys of longer (worse) trees separate these islands

Heuristic Tree Searching
Starting trees can also be obtained via star decomposition
- unlike stepwise addition star decomposition cannot break up neighbors once joined
- this inability to reverse bad decisions causes the performance of star decomposition methods to drop dramatically as the number of OTUs grows
- use as an alternative to stepwise addition to help generate a variety of different starting trees to submit to branch swapping to more thoroughly explore tree space
- Neighbor Joining is a star decomposition method
Heuristic Tree Searching

Star decomposition methods - e.g. Neighbor Joining

Branch Swapping

PAUP allows 3 different types of branch swapping

listed in order of increasing rigor:
- Nearest neighbor interchange (NNI)
- Subtree Pruning and Regrafting (SPR)
- Tree Bisection-Reconnection (TBR)

Branch Swapping

Nearest neighbor interchange (NNI)
Simplest & least thorough branch swapping - don’t use
Tree islands more likely (more likely to get trapped)

Branch Swapping

Subtree Pruning and Regrafting (SPR)
More thorough than NNI - each possible subtree is moved

Branch Swapping

Tree Bisection-Reconnection (TBR)
Most thorough branch swapping procedure
Tree is broken at internal branch & all possible reconnections are made between 2 subtrees

Branch Swapping

TBR swapping is the most thorough method
- Most likely to find better trees, least likely to get trapped on local optima (suboptimal tree islands)
- Also most time consuming

- What if a better tree is found after hours (or days) of swapping?
- Swapping on the worse tree stops and swapping on the better tree begins
- Thus time is saved by starting with as good a tree as possible to reduce the amount of swapping
Branch Swapping

Random Stepwise Addition Sequence Replicates

SUCCESS

FAILURE

Fig. 3. Trace plot of log likelihood for third 3Q GTR + G run that did not reach stationarity until step 2.3 million of a 5-million step run.

Large Datasets

Datasets are getting larger…
- dozens to hundreds, even thousands of OTUs are being analyzed
- famous “test case”
large dataset of Chase et al (1993)
[et al = 41 coauthors!]
dubbed “Zilla”
- 500 angiosperm taxa with rbcL sequences (chloroplast)

Large Datasets: Zilla

Chase et al (1993)
- largest parsimony analysis conducted at the time
- ran for 4 weeks
- found 3,900 trees of length 16,305 steps (16,538 with uninformative characters included)

Large Datasets: Zilla

Rice et al. (1997)
- attempted a reanalysis of Zilla
- strategy included: 3 Sun workstations (UNIX) machines with PAUP 3.0
- 8 heuristic searches with random stepwise addition and TBR swapping
- ran for ~ 11.6 months of CPU time

Large Datasets: Zilla

Rice et al. (1997)
- found 21,774 trees that were shorter than the Chase et al (1993) tree of 16,538
- 8,795 trees of length 16,533 (5 steps shorter than Chase et al)
- examined 27.9 billion trees
- miniscule fraction of total 1.01 x 10^{1280} trees for 500 OTUs
**Large Datasets: Zilla**

Rice et al. (1997)
- Even with PAUP Rice et al could have done much better
- One can tell PAUP to **not** swap trees to completion
  - this allows one to spend less time on each island
  - and allows one to visit more islands
- Rice et al. wasted *months* of time swapping on suboptimal islands
- My experience: a little swapping goes a long way

---

**What Rice et al. (1997) should have done...**

Limit swapping to, say, 5 trees per random addition replicate
19 hours - better trees than Chase et al. (but not better than Rice et al - more time needed) - or used NONA (30x faster than PAUP 3)

---

**Large Datasets: Zilla**


- method to search large datasets
  1. Obtain starting tree & TBR swap
  2. Weight some of the characters (perturb the matrix)
  3. Perform swapping on the best tree(s) of step 1 using the new perturbed matrix
  4. Return weights to original (unperturb matrix)
  5. Take best tree from step 3 for more TBR swapping
  6. Return to step 2 & repeat 200 times

---

**Large Datasets: Zilla**


- For Zilla shorter trees than the Rice et al (1997) trees were found in ~22 hours using the Ratchet

---

**Large Datasets: Zilla**


- Non-ratchet methods do not progress from replicate to replicate
- each replicate is independent & uses no information gained in past replicates
- can take all best trees found in Ratchet search and ‘seed’ them as starting trees for swapping to find all equally optimal topologies
Large Datasets: Zilla


- A variety of search strategies even faster than the Ratchet
- Sectorial Searches, Tree-drifting, Tree-fusing
- Find the shortest Zilla tree in under 10 minutes
- Software: TNT - Tree Analysis Using New Technology

Large Datasets: Zilla

Issues of concern:
- Is a tree of length 16531 actually a better estimate of the phylogeny than a tree of length 16533?
- Cladistic answer: the shorter tree has less homoplasy and is thus the least falsified hypothesis of relationships
- Statistical Phylogenetics answer: probably not - let's compare the two trees statistically to see if one fits the data significantly better than the other

Large Datasets

Issues of concern:
- There has been a shift away from seeking the single optimal tree
- Based on the logic that for most large datasets it is very unlikely we could actually find the single correct tree, ie we will almost certainly be at least a little wrong
- Greater emphasis on representing the strength of the signal in the data
- e.g. consensus trees, bootstrapping, Bayesian Inference (see future lectures)

Terms - from lecture & readings

<table>
<thead>
<tr>
<th>Additive, additive data</th>
<th>Estimated distance data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch swapping</td>
<td>&quot;Swapped to completion&quot;</td>
</tr>
<tr>
<td>Hill climbing / greedy algorithms</td>
<td>Global optima</td>
</tr>
<tr>
<td>Local optima</td>
<td>Stepwise addition</td>
</tr>
<tr>
<td>Ais</td>
<td>Closest</td>
</tr>
<tr>
<td>Simple</td>
<td>Random</td>
</tr>
<tr>
<td>Star decomposition</td>
<td>NNI</td>
</tr>
<tr>
<td>SPR</td>
<td>TBR</td>
</tr>
<tr>
<td>Zilla</td>
<td>Parsimony ratchet</td>
</tr>
</tbody>
</table>

Study questions

If we had a perfect record of every genetic event that occurred in the evolution of two species what sort of distance data would these be?

What are the two basic steps to conducting a heuristic search?

A peer tells you they plan on doing 500 replicates of heuristic searching using a stepwise addition strategy and indicate they'll obtain their starting trees via the ASIS method in PAUP - what is wrong with this strategy?

What role might NJ play in a heuristic search strategy?

Why were Rice et al (1997) only able to complete 8 random addition replicates during their 11+ months of analysis with the Zilla dataset?

Describe a rigorous heuristic search strategy - indicate 1) how the starting tree is obtained 2) what sort of branch swapping is done 3) other ideas?

Describe how the Parsimony Ratchet works (not “why” it works - which no one really knows!)

What is the “shift” away from finding the single optimal tree about?