Effects of Rooting via Outgroups on Ingroup Topology in Phylogeny

Margareta Ackerman*[1], Daniel Brown[2], and David Loker[2]

[1]Department of Computer Science and Engineering, UC San Diego, CA [2]Cheriton School of Computer Science, Waterloo, Ontario

Email: Margareta Ackerman*- maackerman@ucsd.edu; Daniel Brown - browndg@uwaterloo.ca; David Loker - dloker@uwaterloo.ca;

*Corresponding author

Abstract

Users of phylogenetic methods require rooted trees, because the direction of time depends on the placement of the root. Phylogenetic trees are typically rooted through the use of an outgroup. However, this rooting mechanism is inappropriate when adding an outgroup yields a different topology for the ingroup.

We perform a formal analysis of the response of different phylogenetic algorithms to the inclusion of distant outgroups. We prove that linkage-based algorithms, which include UPGMA, do not modify the topology of the ingroup when an outgroup is included. A class of bisecting algorithms are similarly unaffected. These results are the first to provide formal guarantees on the use of outgroups for rooting phylogentic trees, guaranteeing that this rooting mechanism will not effect the structure of any ingroup when certain algorithms are used.

By contrast, the popular neighbour joining algorithm fails this property in a strong sense. Every data set can have its structure destroyed by some arbitrarily distant outlier. Moreover, including multiple outliers can lead to an arbitrary topology on the ingroup. The standard rooting approach that uses outgroups may be fundamentally unsuited for neighbour joining.

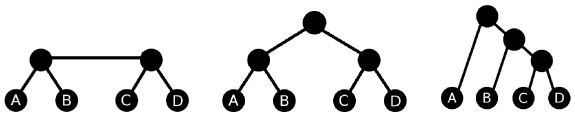
1 Introduction

We perform a theoretical analysis on the role of outgroups in the reconstruction of phylogenetic trees. Consider, for instance, a phylogenetic tree that includes species of seaweed and of bees. The local phylogenies of seaweed and of bees are well separated, and seaweed has no effect on how different types of bees are related to each other. More generally, when sub-trees of the tree of life are sufficiently well separated, they should

have no effect on each others' internal structure.

Virtually all phylogenetics users are ultimately interested in rooted phylogenies, because the direction of time depends on the placement of the root [17]. Rooting is well-known to be "the most precarious step of any phylogenetic analysis" [21]. Incorrect rooting can lead to incorrect conclusions.

Consider, for example, a topology on four taxa where the correct unrooted tree is as shown in Figure 1a (the quartet $AB|CD$) and the correct rooting is as shown in Figure 1b. Now consider an incorrect rooting, as shown in Figure 1c. This interpretation requires an ancestor that led to $\{B, C, D\}$ but not to A, while the correct topology implies that such an ancestor never existed. The incorrect rooting also does not show evidence of an ancestor of only $\{A, B\}$, which is present in the correct rooted topology.



(a) (b) (c)

Figure 1: (a) The correct unrooted toplogy for the taxa A, B, C, D. (b) The correct rooted topology. (c) An incorrect rooted topology.

Outgroups are used for rooting phylogenetic trees ([12, 19]). The standard approach for rooting using outgroups involves adding one or more distant elements, constructing a topology of the ingroup with the added data, and placing the root on the edge where these data connect to the ingroup. It is implicitly assumed that if the outgroup is sufficiently far then it will not affect the structure of the ingroup.

A further problem can occur when adding an outgroup changes the structure of the ingroup. Then, instead of showing how to root the original topology, this rooting mechanism ends up suggesting how to root an entirely different topology.

Holland et. al. [12] found through empirical study that adding an outgroup often disrupts the ingroup structure with neighbour-joining, but not UPGMA. Other authors have also noticed that outgroups can disrupt ingroup topology ([14,19,20]). Our work provides theoretical grounds for these empirical observations.

We explore which algorithms retain ingroup topology when distant outgroups are included (a property we call "outgroup independence"), and which algorithms can disrupt ingroup topology upon the inclusion of arbitrarily distant outgroups. We show that linkage-based algorithms, such as UPGMA, are outgroup independent. We also present a class of outgroup-independent bisecting algorithms. These results provide formal guarantees that when these algorithms are used, the structure of any ingroup will remain intact as long as the outgroup is sufficiently distant.

On the other hand, we show that neighbour joining is not outgroup independent in a strong sense. Every data set can have its structure destroyed by some arbitrarily distant outlier. That is, no matter how far away the outgroup is, the ingroup topology can change by outgroup introduction. We also show that the addition of multiple distant outliers can lead to an arbitrary topology on the ingroup.

2 Previous work

This work falls within the larger framework of the formal study of clustering. Hierarchical algorithms constitute one of the most popular classes of clustering algorithms, and phylogenetic algorithms fall within this class. Recently, Ben-David and colleagues ([2–4, 22]) have proposed a new framework for selecting a clustering algorithm. This approach involves identifying significant properties that distinguish between clustering algorithms in terms of their input/output behavior. When such properties are relevant to the domain knowledge, they may be used to select which algorithms are appropriate. See Ackerman at al. [4] for a detailed exposition of the approach. For example, linkage-based algorithms, one of the most commonly-used classes of hierarchical algorithms, have been shown to be characterized in terms of three intuitive properties ([1, 3]).

Properties of hierarchical algorithms have also been studied in the phylogeny community. A focus of this work has been the classification of algorithms based on their behaviour on additive and ultrametric data or approximations to these properties ([7, 10, 11, 16]). A related line of work is concerned with properties of consensus functions, which provide a way to summarize the agreement between two or more trees, by Bryant [6] and by Eulenstein [9], among others. Lastly, our results in Section 8 bear resemblance to the work of Cueto and Matsen [8] on the BME algorithm. However, our results were obtained independently and we provide simpler proofs.

3 Preliminaries

We consider a general type of input to hierarchical algorithms. A *distance function* is a symmetric function $d : X \times X \to R^+ \cup \{0\}$, such that $d(x, x) = 0$ for all $x \in X$. The data sets that we consider are pairs (X, d), where X is some finite domain set and d is a distance function over X. These are the inputs for distance-based hierarchical algorithms.

A *k-clustering* $C = \{C_1, C_2, \ldots, C_k\}$ of a data set X is a partition of X into k disjoint subsets of X (so, $\cup_i C_i = X$). A *clustering* of X is a k-clustering of X for some $1 \leq k \leq |X|$.

For a clustering C, let $|C|$ denote the number of clusters in C and $|C_i|$ denote the number of points

in cluster C_i. For $x, y \in X$ and a clustering C of X, we write $x \sim_C y$ if x and y belong to the same cluster in C and $x \not\sim_C y$, otherwise. Given a distance function d over X, and $X_1, X_2 \subseteq X$, let $d(X_1, X_2) = \min_{x_1 \in X_1, x_2 \in X_2} d(x_1, x_2)$.

At times, we consider a domain subset with the distance induced from the full domain set. For distance function d over X, and $X' \subseteq X$, $d' = d|X'$ is defined by restricting the distance function d to $X' \times X'$.

Dendrograms are rooted trees whose leaves are associated with elements of the domain. Formally,

Definition 1 (Dendrogram)**.** *A* dendrogram *of X is a pair $D = (T, M)$ where T is a rooted binary tree and $M : leaves(T) \to X$ is a bijection.*

We would like to refer to clusters and clusterings in a dendrogram. Given a dendrogram $\mathcal{D} = (T, M)$ of X, we define a mapping $\mathcal{C} : V(T) \to 2^X$ (where $V(T)$ refers to the set of nodes in T), which takes a node and maps it to the set of all its leaf descendants. Specifically, $\mathcal{C}(x) = \{M(y) \mid y \text{ is a leaf and a descendant of } x\}$. We say that $A \subseteq X$ is *a cluster in $\mathcal{D}$* if there exists a node $x \in V(T)$ so that $\mathcal{C}(x) = A$. We say that a clustering $C = \{C_1, \ldots, C_k\}$ of X is a *clustering in $\mathcal{D}$* if the cluster C_i is in $\mathcal{D}$ for all $1 \leq i \leq k$.

Although we focus on rooted trees, these roots are not always meaningful. These initial roots can be artifacts of the algorithms, and not necessarily the true roots. For example, even though neighbour joining can be said to output a rooted tree, with the root corresponding to the final join operation, it should be treated as unrooted. The last merge is not necessarily between two sub-trees of the true root. We return to the question of whether the roots produced by certain algorithms are meaningful at the end of Section 5 and Section 9.

Definition 2 (Hierarchical algorithm)**.** *A* hierarchical algorithm *$\mathcal{A}$ is a function that takes as input a pair (X, d) and outputs a dendrogram of X.*

In the definition that follows, we say that a node v of degree two with neighbours $\{a, b\}$ is *contracted* if the path $\{a, v, b\}$ is replaced by the edge (a, b).

Definition 3 (Sub-dendrogram)**.** *A dendrogram over $\mathcal{D}' = (T', M')$ is a* sub-dendrogram *of $\mathcal{D} = (T, M)$ if there exists a sub-tree T^* of T such that*

1. *There exists a bijection $\phi : leaves(T^*) \to leaves(T')$ so that $M(x) = M'(\phi(x))$ for all $x \in leaves(T^*)$, and*
2. *T^* equals T after repeatedly contracting all degree 2 vertices in each of T' and T^*, until all nodes in these two trees are of degree 3 or 1.*

Given a dendrogram D of X, and $X' \subseteq X$, the dendrogram D *restricted* to X' is the dendrogram D' over X that is a sub-dendrogram of D.

4 Outgroup independence and outgroup volatility

In this section, we introduce the two central definitions of this paper, namely, outgroup independence and outgroup volatility. Intuitively, an algorithm is outgroup-independent if, whenever an outgroup is sufficiently distant from an ingroup, then the inclusion of the outgroup has no effect on ingroup topology.

Outgroup independence may appear so natural that we might presume all algorithms to be outgroup independent. In fact, neighbour-joining fails this property, while many algorithms, including UPGMA, satisfy it.

To formally define outgroup independence, we first define what it means for a subset of data to be unaffected by another subset.

Definition 4 ($(\mathcal{A}, d)$-unaffected)**.** *Given a distance function d over data set $X \cup O$, X is* $(\mathcal{A}, d)$-unaffected *by O if $\mathcal{A}(X, d)$ is a sub-dendrogram of $\mathcal{A}(X \cup O, d)$, for hierarchical algorithm $\mathcal{A}$.*

A data set X is $(\mathcal{A}, d)$-*affected* by O if it is not $(\mathcal{A}, d)$-unaffected by O.

We now introduce the notion of outgroup independence.

Definition 5 (Outgroup independence)**.** *A hierarchical algorithm $\mathcal{A}$ is* outgroup independent *if for any data sets (X, d) and (O, d'), there exists $c \in R^+$, so that for any d^* over $X \cup O$ that extends both d and d' where $d^*(X, O) \geq c$, X is $(\mathcal{A}, d^*)$-unaffected by O.*

Of course, an algorithm may be outgroup independent but still yield incorrect roots when outgroups are used to root a tree. That placing an outgroup did not disturb ingroup topology does not automatically imply that the root was correctly placed, nor that the topology is correct.

However, outgroup independence is necessary for reliably using outgroups for rooting. If, no matter how far away the outgroup is, the ingroup topology can change by outgroup introduction, then the rooting step is inherently unreliable.

Now we define outgroup volatility in the following strong sense: for any data set, there exist arbitrarily far outgroups that destroy the topology of the ingroup. This is stronger than the opposite of outgroup independence.

Definition 6 (Outgroup/Outlier volatile)**.** *A hierarchical algorithm $\mathcal{A}$ is* outgroup volatile *if, that for any large enough data set (X, d), and any $c \in R^+$, there exists a data set (O, d') and a distance function d^* over $X \cup O$ so that*

1. *d^* extends both d and d'*
2. *$d^*(X, O) \geq c$*
3. *X is $(\mathcal{A}, d^*)$-affected by O.*

If for any (X, d), this definition can be satisfied for a singleton set O, then $\mathcal{A}$ is also outlier volatile*.*

5 Linkage-based algorithms are outgroup independent

In this section we describe a large family of outgroup-independent hierarchical algorithms. In the following two sections, we rely on this result to show that linkage-based algorithms, that include UPGMA, are outgroup independent, and describe a class of outgroup-independent bisecting algorithms.

The large family of outgroup-independent algorithms that we are interested in can be described using three natural properties.

5.1 Properties

5.1.1 Richness

Outer-richness requires that every partition of the domain can be a clustering in the dendrogram produced by the algorithm, for some setting of between-cluster distances. For our purposes, a relaxation is sufficient, making this requirement on pairs of domain sets.

Definition 7 (2-Outer-Richness)**.** *A hierarchical algorithm $\mathcal{A}$ is* 2-outer rich*, if for any (X_1, d_1) and (X_2, d_2), there exists a distance function $\hat{d}$ over $X_1 \cup X_2$ that extends both d_1 and d_2, so that the children of the root of $\mathcal{A}(X_1 \cup X_2, \hat{d})$ are X_1 and X_2.*

5.1.2 Outer-consistency

Outer-consistency is motivated by the idea that increasing between-cluster distances can only improve the certainty of a partition: if we increase between-cluster distances in some clustering, this clustering will also occur in the dendrogram that the algorithm produces on the resulting data.

A distance function d' over X is (C,d)-*outer consistent* if $d'(x,y) = d(x,y)$ whenever $x \sim_C y$, and $d'(x,y) \geq d(x,y)$ whenever $x \not\sim_C y$.

Definition 8 (Outer-consistency)**.** *A hierarchical algorithm $\mathcal{A}$ is* outer consistent *if for any clustering C of X in $\mathcal{A}(X,d)$, if d' is (C,d)-outer consistent, then C is also a clustering in $\mathcal{A}(X,d')$.*

Here is a relaxation of outer-consistency, requiring that the condition hold only for the clustering consisting of the children of the root.

Definition 9 (Weak outer-consistency)**.** *A hierarchical algorithm $\mathcal{A}$ is* weakly outer consistent *if, given that the two children of the root of $\mathcal{A}(X,d)$ are A and B, whenever d' is $(\{A,B\},d)$-outer consistent, then A and B are the two children of the root of $\mathcal{A}(X,d')$.*

Note that outer-consistency implies weak outer-consistency.

5.1.3 Locality

Locality requires that if we select a sub-dendrogram from the dendrogram produced by a hierarchical algorithm, and then produce a hierarchy for the data underlying this sub-dendrogram, then the resulting topology should be consistent with that found in the original sub-dendrogram.

Definition 10 (Locality)**.** *A hierarchical algorithm $\mathcal{A}$ is* local *if for any data set (X,d) and any cluster $Y \in \mathcal{A}(X,d)$, $\mathcal{A}(Y,d|Y)$ is a sub-dendrogram of $\mathcal{A}(X,d)$.*

Variations of the Outer-consistency, Locality, and Richness properties appear in [3], [4], [22], and [13] for the partitional clustering setting, where an algorithm outputs a single partition of the data instead of a dendrogram. Similar properties for the hierarchical setting were proposed in [1]. Our properties are significant relaxations of those in [1].

5.2 A class of outgroup-independent algorithms

We present our general result in this section, showing that all algorithms that satisfy three natural properties are outgroup independent.

Theorem 1. *Any hierarchical algorithm $\mathcal{A}$ that is 2-outer-rich, weakly outer-consistent, and local, is outgroup independent. In addition, for any ingroup X and outgroup O, $\{X,O\}$ is a clustering in $\mathcal{A}(X \cup O,d)$ whenever $d(X,O)$ is sufficiently large.*

Proof. Consider any data sets (X, d) and (O, d'). Since $\mathcal{A}$ is 2-outer-rich, there exists a distance function d^* over $X \cup O$ that extends d and d' so that $\{X, O\}$ is a clustering in $\mathcal{A}(X \cup O, d^*)$. In particular, X and O are children of the root in $\mathcal{A}(X \cup O, d^*)$, since it is a 2-partition of the domain. Let $c = \max_{x \in X, y \in O} d^*(x, y)$.

Let $\hat{d}$ be any distance function where

1. for all $x, y \in X$, $\hat{d}(x, y) = d(x, y)$,
2. for all $x, y \in O$, $\hat{d}(x, y) = d'(x, y)$,
3. for all $x \in X$, $y \in O$, $\hat{d}(x, y) \geq c$,

So $\hat{d}$ is an $(\{X, O\}, d^*)$-outer-consistent distance function over $X \cup O$. Since $\mathcal{A}$ is weakly outer-consistent, we have that $\{X, O\}$ is a clustering in $\mathcal{A}(X \cup O, \hat{d})$.

Since $\mathcal{A}$ is local and $X \in \mathcal{A}(X \cup O, \hat{d})$, $\mathcal{A}(X, d)$ is a sub-tree of $\mathcal{A}(X \cup O, \hat{d})$. □

The latter part of Theorem 1, which requires that $\{X, O\}$ is a clustering in $\mathcal{A}(X \cup O, d)$, implies that algorithms that satisfy the three properties root trees in a manner that is consistent with the rooting that would be obtained using any sufficiently distant outgroup.

6 A class of outgroup-independent linkage-based algorithms

In this section, we show that linkage-based algorithms, a large class of hierarchical algorithms, are outgroup independent. A linkage function looks only at two clusters at a time, and satisfies a few reasonable conditions, while a linkage-based algorithm always merges the two clusters that minimize the linkage function.

A linkage-based hierarchical algorithm relies on a linkage function that determines which clusters are merged at every step of the algorithm. Linkage functions satisfy natural properties, originally presented by Ackerman et al. [3].

Definition 11 (Linkage function)**.** *A* linkage function *is a function*

$$\ell : \{(X_1, X_2, d) \mid d \text{ is over } X_1 \cup X_2\} \to R^+ \cup \{0\}$$

such that,

1. ℓ *is* monotonic*: For all (X_1, X_2, d), if d' is a distance function over $X_1 \cup X_2$ such that for all $x \sim_{\{X_1, X_2\}} y$, $d(x, y) = d'(x, y)$ and for all $x \not\sim_{\{X_1, X_2\}} y$, $d(x, y) \leq d'(x, y)$ then $\ell(X_1, X_2, d') \geq \ell(X_1, X_2, d)$.*
2. *Any pair of clusters can be made arbitrarily distant: For any pair of data sets (X_1, d_1), (X_2, d_2), and any $r \geq 0$, there exists a distance function d that extends d_1 and d_2 such that $\ell(X_1, X_2, d) > r$.*

Let $parent(x)$ be the parent of node x. If x has no parent, then $parent(x) = \emptyset$.

Definition 12 (Linkage-based algorithm)**.** *A hierarchical clustering algorithm $\mathcal{A}$ is* linkage-based *if there exists a linkage function ℓ so that $\mathcal{A}(X, d) = \mathcal{D}$ can be constructed as follows.*

1. *Create $|X|$ nodes, one for every elements of X.*
2. *Let $\{x, y\} = \arg\min_{\{x,y\}\subseteq\mathcal{D}}\{\ell(\mathcal{C}(x), \mathcal{C}(y), d) \mid parent(x) = parent(y) = \emptyset\}$. Then add a node z to T, set $parent(x) = parent(y) = z$, and $parent(z) = \emptyset$.*
3. *Repeat step 2 until a single tree remains.*

Many distance-based hierarchical algorithms are linkage-based algorithms:

- *UPGMA:* $\ell(A, B, d) = \frac{\sum_{a\in A, b\in B} d(a,b)}{|A|\cdot|B|}$
- *Single linkage:* $\ell(A, B, d) = \min_{a\in A, b\in B} d(a, b)$.
- *Complete linkage:* $\ell(A, B, d) = \max_{a\in A, b\in B} d(a, b)$.

We show that all linkage-based algorithms satisfy 2-outer-richness, weak outer-consistency, and locality. By Theorem 1, they are outgroup-independent.

Lemma 1. *Linkage-based algorithms are 2-outer rich.*

Proof. Let (X_1, d_1) and (X_2, d_2) be some data sets. We will show that there exists an extension d of d_1 and d_2 so that $\{X_1, X_2\}$ is a clustering in $\mathcal{A}(X_1 \cup X_2, d)$.

To make $\mathcal{A}$ give this output, we design d in such a way that for any $i \in \{1, 2\}$, and $A, B \subseteq X_i$, and any $C \subseteq X_1$ and $D \subseteq X_2$, $\ell(A, B, d) < \ell(C, D, d)$, which means that the first time we join any members of X_1 with any members of X_2 will be the last merging, when X_1 and X_2 are merged.

Let $r = \max_{i\in\{1,2\}, A,B\subseteq X_i} \ell(A, B)$. Since ℓ satisfies property 2 of Definition 11, for any $C \subseteq X_1$, $D \subseteq X_2$, there exists a distance function d_{CD} that extends d_1 and d_2 so that $\ell(C, D) > r$. Consider constructing such distance function d_{CD} for every pair $C \subseteq X_1$ and $D \subseteq X_2$. Then, let $m = \max_{C\subseteq X_1, D\subseteq X_2} \max_{x\in C, y\in D} d_{CD}(x, y)$.

We define d as follows: $d(x, y) = d_1(x, y)$ if $x, y \in X_1$, $d(x, y) = d_2(x, y)$ if $x, y \in X_2$, and $d(x, y) = m$ otherwise. Since ℓ satisfies property 1 of Definition 11, $\ell(C, D, d) > r$ for all $C \subseteq X_1$, $D \subseteq X_2$. On the other hand, $\ell(A, B, d) \leq r$ for any $A, B \subseteq X_1$ or $A, B \subseteq X_2$. Therefore, the algorithm will not merge any $C \subseteq X_1$

with $D \subseteq X_2$, while there is any clusters $A, B \subseteq X_1$ or $A, B \subseteq X_2$ remaining. This gives that $\{X_1, X_2\}$ is a clustering in $\mathcal{A}(X_1 \cup X_2, d)$.

□

Lemma 2. *Linkage-based algorithms are weakly outer consistent.*

Proof. Consider any linkage-based algorithm with linkage function ℓ. For any data set (X, d), let A and B be the clusters corresponding to the children of the root X, the elements that were the last to merge. Now consider some $(\{A, B\}, d)$-outer-consistent distance function d'. Then for any $A' \subseteq A$, and any $B' \subseteq B$, $\ell(A', B', d') \geq \ell(A', B', d)$ by monotonicity (Def 11 part 1). On the other hand, for any $A_1, A_2 \subseteq A$, $\ell(A_1, A_2, d') = \ell(A_1, A_2, d)$. Similarly, for any $B_1, B_2 \subseteq B$, $\ell(B_1, B_2, d') = \ell(B_1, B_2, d)$. Therefore, on input (X, d'), clusters A and B are still formed, and so they are the children of the root. □

Lemma 3. *Linkage-based algorithms are local.*

Proof. By way of contradiction, assume that some linkage-based algorithm $\mathcal{A}$ that is not local. Then there is some data set (X, d) and $X' \subseteq X$, where X' is a node in $\mathcal{A}(X, d)$, so that $\mathcal{A}(X', d|X')$ is not a sub-dendrogram of (X, d). Then at some point during the execution of $\mathcal{A}$ on $(X', d|X')$ some nodes representing clusters Y' and Z' are merged, which are not merged in the execution of $\mathcal{A}$ on $(X, d|X)$. Say (Y', Z') is the earliest pair that merges in $(X', d|X')$ but not in $(X, d|X)$. But then the next time that a pair of subsets of X' are merged in the execution of $\mathcal{A}$ on (X, d) it must be Y' and Z', since their merging in the execution of $\mathcal{A}$ on $(X', d|X')$ indicates that their $\ell(Y', Z', d/Y' \cup Z')$ is minimal over all current pairs of subsets of X'. So no such Y' and Z' exist. □

Theorem 2. *Linkage-based algorithms are outgroup independent. In addition, for any ingroup X and outgroup O, $\{X, O\}$ is a clustering in $\mathcal{A}(X \cup O, d)$ whenever $d(X, O)$ is sufficiently large.*

Proof. The result follows by Theorem 1, Lemma 1, Lemma 2, and Lemma 3. □

Not only are linkage-based algorithms outgroup independent, but the root that they find without an outgroup is also the root that they would find with any sufficiently distant outgroup. That is, if we use any linkage-based algorithm and treat the output as an unrooted tree, any sufficiently distant outgroup will find the same root as the algorithm would have found.

Since UPGMA, single-linkage, and complete-linkage are linkage-based algorithms, the following corollary holds.

Corollary 1. *UPGMA, single-linkage, and complete-linkage are outgroup independent.*

7 Outgroup-independent bisecting algorithms

We define another class of outgroup-independent hierarchical algorithms, which includes the popular bisecting k-means algorithm.

A 2-*clustering* function $\mathcal{F}$ is a function which, given a data set (X, d), outputs a 2-clustering of X. The $\mathcal{F}$-Bisecting hierarchical algorithm constructs a dendrogram in a top-down manner. It begins by creating a dendrogram with a single node that represents the entire data set. It then uses $\mathcal{F}$ on the entire data set, and assigns the resulting clusters to be the children of the root. The algorithm repeatedly applies $\mathcal{F}$ on leaves that represent clusters of size at least two, until no such leaves remain.

Definition 13 ($\mathcal{F}$-Bisecting)**.** *Given a 2-clustering function $\mathcal{F}$, the $\mathcal{F}$-Bisecting hierarchical algorithm, on input (X, d), returns the dendrogram (T, M) where for any node $x \in V(T)/\text{leaves}(T)$ with children x_1 and x_2, $\mathcal{F}(\mathcal{C}(x)) = \{\mathcal{C}(x_1), \mathcal{C}(x_2)\}$.*

At the end of this section, we show that the set of $\mathcal{F}$-Bisecting algorithms is not the same as the set of linkage-based algorithms.

We now show that if $\mathcal{F}$ satisfies some reasonable conditions, then $\mathcal{F}$-bisecting is outgroup independent. Consider the following two properties of 2-clustering functions.

Definition 14 (Outer richness for 2-clustering functions)**.** *A 2-clustering function $\mathcal{F}$ is* outer rich *if for every set of domains, (X_1, d_1) and (X_2, d_2), there exists a distance function $\hat{d}$ over $X_1 \cup X_2$ that extends d_1 and d_2, such that $\mathcal{F}(X_1 \cup X_2, \hat{d}) = \{X_1, X_2\}$.*

Definition 15 (Outer consistency for 2-clustering functions)**.** *A 2-clustering function $\mathcal{F}$ is* outer consistent *if for every (X, d), if d' is $(\mathcal{F}(X, d), d)$-outer consistent then $\mathcal{F}(X, d) = \mathcal{F}(X, d')$.*

Theorem 3. *For any outer-consistent and outer-rich 2-clustering function $\mathcal{F}$, the $\mathcal{F}$-bisecting algorithm is outgroup independent.*

Proof. Since $\mathcal{F}$ is outer-rich, $\mathcal{F}$-bisecting is 2-outer-rich. Since $\mathcal{F}$ is outer-consistent, $\mathcal{F}$-bisecting is weakly outer-consistent, since the first split is just the output of $\mathcal{F}$. Finally, every $\mathcal{F}$-bisecting algorithm is local, which follows directly from the top-down construction of bisecting algorithms. Therefore, by Theorem 1, $\mathcal{F}$-bisecting is outgroup independent.

□

We refer to 2-means-bisecting with its more common name "bisecting k-means." The k-means function finds a k-clustering that minimizes the average distance of an element to the center-of-mass of its cluster. Bisecting k-means has been referred to as the "best representative of the class of iterative centroid-based bisecting algorithms" [18].

Corollary 2. *The hierarchical algorithm bisecting k-means is outgroup independent.*

Proof. Ackerman et al. [4] show that k-means is outer-consistent and outer-rich, so in particular these conditions are satisfied when $k = 2$. The result follows by Theorem 3.

□

Although k-means is outer-consistent, bisecting k-means is not. This property is not required for Theorem 3. By contrast, all linkage-based algorithms are outer consistent [1]. This shows that the class of bisecting algorithm is distinct from the class of linkage-based algorithms.

Lemma 4. *Bisecting k-means is not outer consistent.*

Proof. The k-means objective function is

$$k\text{-means}(C, X, d) = \sum_{C_i \in C} \frac{1}{|C_i|} \sum_{x,y \in C_i} d(x, y)^2.$$

Note that this definition is equivalent to the more well-known formulation which relies on centers of mass [15]. Consider a data set (X, d) with $X = \{1, 2, \ldots, 16n\}$ and $d(i, j) = |i - j|$. In the tree produced by bisecting k-means, let A and B be the clusters represented by the children of the root, A_1 and A_2 the children of A, and B_1 and B_2 the the children of B. Then $\{A_1, A_2, B_1, B_2\}$ is a clustering in the dendrogram produced by bisecting k-means on (X, d).

It can easily be shown that clusters in this clustering have $4n$ elements. However, if we move B_2 sufficiently far from $A \cup B_1$, we are still $(\{A_1, A_2, B_1, B_2\}, d)$-outer consistent, but the children of the root of the tree produced by bisecting k-means on the new data becomes $A \cup B_1$ and B_2, and $A \cup B_1$ is split into two clusters each of size $6n$, and each is then further divided into clusters of size $3n$. So clusters A_1 and A_2, of size $4n$, are never formed. Therefore $\{A_1, A_2, B_1, B_2\}$ is not a clustering in the new tree, so the algorithm is not outer consistent. □

8 Neighbour joining is outgroup volatile

We show that neighbour joining (NJ) is outgroup volatile in a strong sense; even a single, arbitrarily far outlier can destroy the topology of an ingroup. Thus, we cannot eliminate such behaviour by increasing the

outlier's distance from the ingroup.

We then show that we can completely destroy the tree structure of the ingroup by adding multiple outliers. That is, we will show that for *every* topology of the ingroup, there is a set of arbitrarily distant outliers that can lead to this topology when using neighbour joining. These results are similar to those of Cueto and Matsen [8] for the BME algorithm, though our proof for NJ is independent of theirs and more elementary.

Neighbour joining is an iterative algorithm. Each iteration consists of the following steps, for n nodes:

1. Based on the current distance matrix, calculate the matrix Q as follows:
   - $Q(i,j) = (n-2)d(i,j) - \sum_k d(i,k) - \sum_k d(j,k)$
2. Find the pair of taxa, A and B, in Q with the lowest value. Create a node U on the tree that joins these two taxa.
3. Calculate the distances $d(A,U)$ and $d(B,U)$ as follows:
   - $d(A,U) = \frac{1}{2}d(A,B) + \frac{1}{2(n-2)}[\sum_k d(A,k) - \sum_k d(B,k)]$
   - $d(B,U) = \frac{1}{2}d(A,B) + \frac{1}{2(n-2)}[\sum_k d(B,k) - \sum_k d(A,k)]$
4. Calculate the distances $d(U,k)$, for all $k \notin \{A,B\}$ as follows:
   - $d(U,k) = \frac{1}{2}[d(A,k) + d(B,k) - d(A,U) - d(B,U)]$
5. Repeat the algorithm again, considering the pair of joined neighbours as a single taxon and using the distances calculated in the previous step. Stop when there is a single taxon remaining.

Theorem 4. *Neighbour joining is outlier volatile.*

Proof. Consider any data set (X,d) and an outgroup consisting of a single element, $X' = \{O\}$. Let c be any real number greater than $\sum_{x,y \in X} d(x,y)$. Let d^* be a distance function over $X \cup X'$ that extends d.

Pick any clusters A and B in X that are not cherries in $NJ(X,d)$. We will set the distances between O and X so that X is (NJ, d^*)-affected by X'; specifically, in the tree that X induces on $NJ(X \cup X', d^*)$, A and B will be cherries.

For d^*, we now define the distances between X and X', as follows: $d^*(A,O) = c$, $d^*(B,O) = c$, and for all $k \in X$, $k \notin \{A,B\}$, set $d^*(k,O) = 10c$.

We show that in the first step of neighbour joining, O attaches to one of A or B (call the resulting node U). At the next step, neighbour joining attached U with whichever of A or B remains (this is shown below). Since A and B are merged before anything else in the dendrogram, the sub-dendrogram of $NJ(X \cup X', d^*)$ containing only those taxa in X has A and B as cherries. Thus, X is (NJ, d^*)-affected by X'.

It remains to show that after O attaches to one of A or B, neighbour joining attached U with whichever of A or B remains.

We need to evaluate the value of $Q(k, O)$, for all $k \in X$, $k \notin \{A, B\}$. We assume that the number of points in our data set $X \cup X'$ is n.

$$\begin{aligned}
Q(k,O) &= (n-2)d^*(k,O) - \sum_{j \in X \cup X'} d^*(k,j) - \sum_{j \in X \cup X'} d^*(j,O) \\
&= (n-2)10c - d^*(k,O) - \sum_{j \in X} d^*(k,j) - d^*(A,O) \\
&- d^*(B,O) - d^*(O,O) - (n-3)10c \\
&= -2c - \sum_{j \in X} d^*(k,j)
\end{aligned}$$

Next, we evaluate $Q(A, O)$.

$$\begin{aligned}
Q(A,O) &= (n-2)d^*(A,O) - \sum_{j \in X \cup X'} d^*(A,j) - \sum_{j \in X \cup X'} d^*(j,O) \\
&= (n-2)c - d^*(A,O) - \sum_{j \in X} d^*(A,j) - d^*(A,O) \\
&- d^*(B,O) - d^*(O,O) - (n-3)10c \\
&= (n-2-1-1-1-10 \cdot n + 30)c - \sum_{j \in X} d^*(A,j) \\
&= (25-9n)c - \sum_{j \in X} d^*(A,j).
\end{aligned}$$

And $Q(B, O) = (25 - 9n)c - \sum_{j \in X} d^*(B, j)$ similarly. And finally, consider $Q(i, j)$ for all $i, j \in X$.

$$
\begin{aligned}
Q(i,j) &= (n-2)d^*(i,j) - \sum_{k \in X \cup X'} d^*(i,k) - \sum_{k \in X \cup X'} d^*(j,k) \\
&= (n-2)d^*(i,j) - d^*(i,O) - d^*(j,O) - \sum_{k \in X} d^*(i,k) - \sum_{k \in X} d^*(j,k) \\
&> (n-2)d^*(i,j) - 10c - 10c - \sum_{k \in X} d^*(i,k) - \sum_{k \in X} d^*(j,k) \\
&> (n-2)d^*(i,j) - 20c - c - \sum_{k \in X} d^*(j,k) \\
&> (n-2)d^*(i,j) - 20c - c - c \\
&= (n-2)d^*(i,j) - 22c
\end{aligned}
$$

Thus, for all $i, j \in X$, $Q(i,j) > (n-2)d^*(i,j) - 22c$. Assume without loss of generality that $Q(A,O) < Q(B,O)$. When is $Q(A,O)$ the smallest value for any $Q(i,j)$?

$$
\begin{aligned}
Q(A,O) < Q(k,O) \text{ for all } k \in X,\ k \notin \{A,B\} \quad &\iff \\
(25-9n)c - \sum_{j \in X} d^*(A,j) < -2c - \sum_{j \in X} d^*(k,j) \quad &\iff \\
(9n-27)c > \sum_{j \in X} d^*(k,j) - \sum_{j \in X} d^*(A,j) \quad &\Leftarrow \\
(9n-27)c > c \quad &\iff \\
n > \frac{28}{9}
\end{aligned}
$$

$$
\begin{aligned}
Q(A,O) < Q(i,j) \text{ where } i,j \in X \quad &\Leftarrow \\
(25-9n)c - \sum_{j \in X} d^*(A,j) < (n-2)d^*(i,j) - 22c < Q(i,j) \quad &\iff \\
(9n-47)c > -(n-2)d^*(i,j) - \sum_{j \in X} d^*(A,j) \quad &\Leftarrow \\
(9n-47)c > 0 \quad &\iff \\
n > \frac{47}{9}
\end{aligned}
$$

When $n \geq 6$, so $|X| \geq 5$, $Q(A,O)$ and $Q(B,O)$ are smaller than any other $Q(i,j)$. Assume without loss of generality that $Q(A,O)$ is minimal. We create node U by merging A and O. To accomplish this, we must calculate $d^*(A,U)$, $d^*(O,U)$, $d^*(U,B)$ and finally $d^*(U,k)$, for all $k \neq B$. For brevity, we omit the steps and just give the results. We redefine $X = X \setminus \{A\}$ and let $X' = \{U\}$.

Note that, in various places in the calculations that follow, we make reference to $d^*(A,k)$ and $d^*(O,k)$, for some k. We are referring to the distances defined previously for d^* since they will not need to be changed. Thus, for these distances, we refer to the distance matrix from the first iteration of the algorithm.

$$
\begin{aligned}
d^*(A,U) &= \frac{1}{(2(n-2))}[(27-9n)c + \sum_{k\in X} d^*(A,k)] \\
d^*(O,U) &= \frac{1}{(2(n-2))}[(11n-31)c - \sum_{k\in X} d^*(A,k)] \\
d^*(U,B) &= \frac{1}{2}[d^*(A,B) - d^*(A,U)] + \frac{1}{2}[d^*(O,B) - d^*(O,U)] \\
&= \frac{1}{2}d^*(A,B) \\
d^*(U,k) &= \frac{1}{2}d^*(A,k) + \frac{9}{2}c
\end{aligned}
$$

Now, we calculate $Q(U,B)$.

$$
\begin{aligned}
Q(U,B) &= \frac{(n-2)}{2}d^*(A,B) - \sum_{k\in X\cup X'} d^*(U,k) - \sum_{k\in X\cup X'} d^*(B,k) \\
&= \frac{n}{2}d^*(A,B) - \frac{(n-2)9}{2}c - \frac{1}{2}\sum_{k\in X} d^*(A,k) - \sum_{k\in X} d^*(B,k)
\end{aligned}
$$

For all $k \neq B$, we calculate $Q(U,k)$.

$$
\begin{aligned}
Q(U,k) &= \frac{(n-2)}{2}[d^*(A,k) + 9c] - \sum_{j\in X\cup X'} d^*(U,j) - \sum_{j\in X\cup X'} d^*(k,j) \\
&= \frac{(n-3)}{2}d^*(A,k) - \frac{9}{2}c - \frac{1}{2}d^*(A,B) - \sum_{j\in X} d^*(k,j) - \frac{1}{2}\sum_{j\in X} d^*(A,j)
\end{aligned}
$$

So, $Q(U,B) < Q(U,k)$ whenever,

$$
\begin{aligned}
\frac{(n-3)9}{2}c &> \frac{n-1}{2}d^*(A,B) + \sum_{j\in X} d^*(k,j) \\
&\quad - \sum_{j\in X} d^*(B,j) - \frac{(n-3)}{2}d^*(A,k) \quad \Leftarrow \\
9(n-3)c &> (n-1)d^*(A,B) + 2c \quad \Leftarrow \\
9n - 27 &> n + 1 \quad \Longleftrightarrow \\
n &> \frac{28}{8} \quad \Leftarrow \\
n &\geq 4
\end{aligned}
$$

For $i,j \in X$, $i,j \neq U$, we have that:

$$
\begin{aligned}
Q(i,j) &= (n-2)d^*(i,j) - \sum_{k \in X \cup X'} d^*(i,k) - \sum_{k \in X \cup X'} d^*(j,k) \\
&= (n-2)d^*(i,j) - d^*(i,U) - d^*(j,U) - \sum_{k \in X} d^*(i,k) - \sum_{k \in X} d^*(j,k) \\
&> (n-2)d^*(i,j) - \frac{1}{2}d^*(A,i) - \frac{9}{2}c - \frac{1}{2}d^*(A,j) - \frac{9}{2}c - 2c \\
&> (n-2)d^*(i,j) - 12c
\end{aligned}
$$

We want $Q(U,B) < Q(i,j)$, which holds whenever $c(9(n-2)-24) > n \cdot d^*(A,B)$, which holds when $n \geq 6$.

So the next merge is U and B.

□

Note also that all ingroups, even ones that correspond to additive distances, are volatile with respect to arbitrarily distant outgroups. However, the ingroup and outgroup together may not be additive, since then NJ is consistent. That is, if the result of introducing the outgroup must remain additive, then the outgroup cannot disrupt ingroup topology.

In fact, when more outliers are added, neighbour joining can give an arbitrary topology over the original data. This remains the case even when the outliers are arbitrarily far from the ingroup, and when the internal structure of the outgroups is additive. This shows that outliers can completely override the original data, leading to ingroup structure that can be arbitrarily different from that obtained without the outgroup.

Theorem 5. *Given any data set $X = \{X_1, X_2, \ldots, X_m\}$, with distances d over X, there exists a set of outliers $O = \{O_1, O_2, \ldots, O_m\}$ and a distance function d^* over $X \cup O$ extending d, where $d^*(X, O_i)$ can be arbitrarily large for all $1 \leq i \leq m$, such that $NJ(X \cup O, d^*)$ restricted to X results in an arbitrary dendrogram.*

Proof. Consider any dendrogram $D = (T, M)$ over O. Choose any additive distances d' so that $NJ(O, d') = D$; without loss of generality, assume that the minimum distance in d' is 1. For any scaling of d', NJ yields the same dendrogram.

We now construct a distance function d^* that extends d, and that scales d' for pairs of elements of O. We pick two large constants L and t, and let $d^*(O_i, O_j) = 4L \cdot d'(O_i, O_j)$, $d^*(O_i, X_i) = t$, and $d^*(O_i, X_j) = 20t$ if $i \neq j$. With sufficiently large t and L, $d^*(X, O_i)$ is arbitrarily large for all i, and similarly, $d^*(O_i, O_j)$ is arbitrarily large.

We show that there exist L and t such that $NJ(X \cup O, d^*)$ is a dendrogram D' such that:

1. (O_i, X_i) is a cherry with parent U_i,
2. D' restricted to O_i is the dendrogram D,
3. D' restricted to X_i is the dendrogram (T, M'), where $M'(v) = X_i \iff M(v) = O_i$, for all $v \in T$. That is, (T, M') and dendrogram D are the same up to a relabeling of the leaves.

We show that the first m mergings in the execution of NJ are all of O_i and X_i for some ordering of the m pairs, and that then, the algorithm follows the same mergings as in the execution $NJ(O, d')$.

Let $n = 2m$, which is the total number of elements in $O \cup X$. Let $S = \max_{1 \leq i \leq m} \sum_{k \in X} d(X_i, k)$.

First, we show that $Q(O_i, X_i)$ is smaller than both $Q(O_i, X_j)$ and $Q(X_i, X_j)$, for all $i \neq j$.

$$\begin{aligned} Q(O_i, X_i) &= (n-2)d^*(O_i, X_i) - \sum_k d^*(O_i, k) - \sum_k d^*(X_i, k) \\ &= (n-2)t - \sum_k d^*(O_i, k) - \sum_k d^*(X_i, k) \end{aligned}$$

We compare this value to $Q(O_i, X_j)$,

$$\begin{aligned} Q(O_i, X_j) &= (n-2)d^*(O_i, X_j) - \sum_k d^*(O_i, k) - \sum_k d^*(X_j, k) \\ &= (n-2)20t - \sum_k d^*(O_i, k) - \sum_k d^*(X_j, k) \end{aligned}$$

Consider their difference, $Q(O_i, X_j) - Q(O_i, X_i)$ which is $(n-2)19t - \sum_k d^*(X_j, k) + \sum_k d^*(X_i, k)$. All the terms in t that occur in $-\sum_k d^*(X_j, k) + \sum_k d^*(X_i, k)$ cancel out. The remainder, is bigger than $-S$. Thus, for sufficiently large t, the difference is positive. So, O_i would merge with X_i, and not X_j, for $j \neq i$.

Next, we must evaluate if $Q(X_\ell, X_j) - Q(O_i, X_i)$ is positive, for any $\ell \neq j$. We calculate $Q(X_\ell, X_j)$, which is the following:

$$Q(X_\ell, X_j) = (n-2)d^*(X_\ell, X_j) - \sum_k d^*(X_\ell, k) - \sum_k d^*(X_j, k)$$

Thus, for t sufficiently large, the difference $Q(X_\ell, X_j) - Q(O_i, X_i)$ is given by:

$$\begin{aligned}
&Q(X_\ell, X_j) - Q(O_i, X_i) \\
&= (n-2)d^*(X_\ell, X_j) - \sum_k d^*(X_\ell, k) - \sum_k d^*(X_j, k) \\
&\quad - (n-2)t + \sum_k d^*(O_i, k) + \sum_k d^*(X_i, k) \\
&> 0 - n \cdot 20t - n \cdot 20t - (n-2)t + \frac{n}{2} \cdot 4L + 0 \\
&= -(41n-2)t + 2nL
\end{aligned}$$

For L sufficiently large (much larger than t), the difference is positive. Finally, it remains to see if $Q(O_\ell, O_j)$ is larger than $Q(O_i, X_i)$, for any $\ell \neq j$.

$$\begin{aligned}
&Q(O_\ell, O_j) = (n-2)d^*(O_\ell, O_j) - \sum_k d^*(O_\ell, k) - \sum_k d^*(O_j, k) \\
&> (n-2) \cdot 4L - \sum_k d^*(O_\ell, k) - \sum_k d^*(O_j, k) \\
&> (n-2) \cdot 4L - (\frac{n}{2} - 1) \cdot 4L - \sum_{k \in X} d^*(O_\ell, k) \\
&\quad - (\frac{n}{2} - 1) \cdot 4L - \sum_{k \in X} d^*(O_j, k) \\
&= (n - n - 2 + 2) \cdot 4L - \sum_{k \in X} d^*(O_\ell, k) - \sum_{k \in X} d^*(O_j, k) \\
&= - \sum_{k \in X} d^*(O_\ell, k) - \sum_{k \in X} d^*(O_j, k)
\end{aligned}$$

Thus, for sufficiently large L, $Q(O_\ell, O_j) > -2L$. This is clearly larger than $Q(O_i, X_i)$, which contains the term $-\sum_k d^*(O_i, k)$ which is less than $-2nL$.

Thus, NJ first merges O_i and X_i into U_i, for some i.

We show that the node U_i, created after the merge, has similar distances to the all remaining nodes as the distances from O_i. We will see that the largest term in each distance will remain the same. We will also see that if there are two U_i and U_j nodes, $d^*(U_i, U_j)$ has the same order of magnitude as $d^*(O_i, O_j)$. Finally, we will see that $d^*(U_i, X_j)$ will always be larger than $d^*(O_j, X_j)$, for the remaining non-merged O_j, X_j pairs. This will be sufficient for our argument above that $Q(O_j, X_j) < Q(U_i, X_j)$.

Thus, for t and L sufficiently large (much larger than t), our above argument will continue to hold, and NJ will continue to merge O_j with X_j until they are all pairs, and we are left only with U_i nodes, for all i.

First, we calculate $d^*(U_i, O_j)$ for $j \neq i$. Note that $d^*(O_i, U_i) + d^*(X_i, U_i) = d^*(O_i, X_i)$ because the terms that are multiplied by $\frac{1}{2(n-2)}$ cancel out.

$$
\begin{aligned}
d^*(U_i, O_j) &= \frac{1}{2}[d^*(O_i, O_j) + d^*(X_i, O_j) - [d^*(O_i, U_i) + d^*(X_i, U_i)]] \\
&= \frac{1}{2}[d^*(O_i, O_j) + d^*(X_i, O_j) - d^*(O_i, X_i)] \\
&> \frac{1}{2}4L \\
&= 2L
\end{aligned}
$$

Thus, $d^*(U_i, O_j)$ has the same order of magnitude as $d^*(O_i, O_j)$. Next, we calculate $d^*(U_i, X_j)$.

$$
\begin{aligned}
d^*(U_i, X_j) &= \frac{1}{2}[d^*(O_i, X_j) + d^*(X_i, X_j) - d^*(O_i, X_i)] \qquad\qquad &> \frac{1}{2}[20t - t] \\
&= \frac{19}{2}t
\end{aligned}
$$

Thus, $d^*(U_i, X_j) > d^*(O_j, X_j)$. Next, we calculate $d^*(U_i, U_j)$, assuming both have been created by merging their respective O_ℓ, X_ℓ pairs.

$$
\begin{aligned}
d^*(U_i, U_j) &= \frac{1}{2}[d^*(O_i, U_j) + d^*(X_i, U_j) - d^*(O_i, X_i)] \\
&> \frac{1}{2}[2L + d^*(X_i, U_j) - d^*(O_i, X_i)] \\
&> L + \frac{17}{4}t \\
&> L
\end{aligned}
$$

Therefore, $d^*(U_i, U_j)$ has the same order of magnitude as $d^*(O_i, O_j)$. For L sufficiently large, and much larger than t, NJ will continually merge O_ℓ, X_ℓ pairs, until none remain.

We now show that the distances between the U_i's approaches $\frac{1}{4}d^*(O_j, O_i)$ as L grows. Atteson [5] showed that if the distance estimates are at most half the minimal edge length of the tree away from their true, additive, value then neighbour-joining will reconstruct the correct tree. Since the distances on O are additive, and a scaling of additive data is additive, showing that distances between the U_i's approaches $\frac{1}{4}d^*(O_j, O_i)$ as L grows completes the proof.

We need to bound $d^*(U_i, U_j)$ from above. We get the following:

$$
\begin{aligned}
d^*(U_i, U_j) &= \frac{1}{2}[d^*(O_i, U_j) + d^*(X_i, U_j) - d^*(O_i, X_i)] \\
&= \frac{1}{2}[\frac{1}{2}[d^*(O_j, O_i) + d^*(X_j, O_i) - d^*(O_j, X_j)] + d^*(X_i, U_j) - d^*(O_i, X_i)] \\
&= \frac{1}{4}d^*(O_j, O_i) + f(t),
\end{aligned}
$$

where the function $f(t)$ is positive and independent of L. This approximation can be arbitrarily close. The resulting distance matrix over the U_i's is an arbitrarily tight approximation of the original dendrogram D over O.

□

Note that the above theorem applies even when the original data is additive. That is, given additive data, there exist arbitrarily distant outliers (also arbitrarily distant from each other) that completely disrupt the structure of the ingroup.

9 Conclusions

Empirical studies have shown that the inclusion of outgroups may disrupt ingroup topology ([12,14,19,20]). We investigate this issue from a theoretical viewpoint. We ask: is it possible to prevent disrupting the ingroup by moving the outgroup sufficiently far away? We show that this is indeed the case for linkage-based algorithms, such as UPGMA. We also describe a class of bisecting algorithms where this holds. When rooting using distant outliers, these algorithms can be relied on without the risk of disrupting ingroup topology.

On the other hand, we show that for neighbour joining, even a single arbitrarily far outlier can disrupt ingroup topology. Adding multiple outliers, which can be arbitrarily far from the ingroup, can completely destroy the ingroup's toplogy; that is, for every topology of the ingroup, there is a set of outliers that lead to this topology. Further research is necessary to understand if it is possible to find scenarios where NJ is not volatile to the effects of outgroups.

References

1. M. Ackerman and S. Ben-David. Discerning linkage-based algorithms among hierarchical clustering methods. In *IJCAI*, 2011.

2. M. Ackerman, S. Ben-David, S. Branzei, and D. Loker. Weighted clustering. In *AAAI*, 2012.

3. M. Ackerman, S. Ben-David, and D. Loker. Characterization of linkage-based clustering. In *COLT*, 2010.

4. M. Ackerman, S. Ben-David, and D. Loker. Towards property-based classification of clustering paradigms. In *NIPS*, 2010.

5. K. Atteson. The performance of neighbor-joining methods of phylogenetic reconstruction. *Algorithmica*, 25(2):251–278, 1999.

6. D. Bryant. Building trees, hunting for trees and comparing trees. *Unpublished PhD Thesis. Department of Mathematics, University of Canterbury, New Zealand*, 1997.

7. D. Bryant. On the uniqueness of the selection criterion in neighbor-joining. *Journal of Classification*, 22(1):3–15, 2005.

8. M.A. Cueto and F.A. Matsen. Polyhedral geometry of phylogenetic rogue taxa. *Bulletin of Mathematical Biology*, 2010.

9. O Eulenstein. Consensus trees and supertrees. In S. Aluru, editor, *Handbook of computational molecular biology*. CRC Press, 2006.

10. McKenzie A. Gascuel, O. Performance Analysis of Hierarchical Clustering Algorithms. *Journal of Classification*, 21:3–18, 2004.

11. D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Univ Pr, 1997.

12. B.R. Holland, D. Penny, and M.D. Hendy. Outgroup misplacement and phylogenetic inaccuracy under a molecular clock: a simulation study. *Systematic biology*, 52(2):229, 2003.

13. J. Kleinberg. An impossibility theorem for clustering. *Advances in Neural Information Processing Systems*, pages 463–470, 2003.

14. Y.H. Lin, P.J. Waddell, and D. Penny. Pika and vole mitochondrial genomes increase support for both rodent monophyly and glires. *Gene*, 294(1-2):119–129, 2002.

15. R. Ostrovsky, Y. Rabani, L.J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 165–176. IEEE, 2006.

16. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, 1987.

17. M.J. Sanderson and H.B. Shaffer. Troubleshooting molecular phylogenetic analyses. *Annual Review of Ecology and Systematics*, 33:49–72, 2002.

18. S.M. Savaresi and D.L. Boley. On the performance of bisecting K-means and PDDP. *Proceedings of the 1 st SIAM ICDM, Chicago, IL*, 2001.

19. L. Shavit, D. Penny, M.D. Hendy, and B.R. Holland. The problem of rooting rapid radiations. *Molecular biology and evolution*, 24(11):2400, 2007.

20. K.E. Slack, A. Janke, D. Penny, and U. Arnason. Two new avian mitochondrial genomes (penguin and goose) and a summary of bird and reptile mitogenomic features. *Gene*, 302(1-2):43–52, 2003.

21. D.L. Swofford, G.J. Olsen, P.J. Waddell, and D.M. Hillis. Phylogenetic inference. 1996.

22. R.B. Zadeh and S. Ben-David. A uniqueness theorem for clustering. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 639–646. AUAI Press, 2009.