Analyzing the Phylogeny of the Red Panda Using Neighbor Joining and Parsimony Methods

Megan Adams[1], **Sandee Basco**[2], **Bella Falkenberg**[3], **and Alexa Sheldon**[4]

[1]Statistics, Junior, maa367 1
[2]Computer Science, Junior, myb22 2
[3]Statistics and Computer Science, Junior, alf253 3
[4]Computer Science, Junior, ars422 4

ABSTRACT

The Red Panda has been a curiosity as it has been found to be a monotypic taxon, the only member in its family *Ailuridae*, though historically, it was placed in the raccoon family and then the bear family. We utilize mitochondrial DNA from red pandas to analyze and confirm the results of separation of the red panda into two separate species, as outlined in Joshi et al. [1]. Particularly, our motivation to analyze red pandas is to gain insight into what has led to such controversy in its classification. To analyze the phylogeny, we will be using multiple methods of phylogenetic construction. We implement a Weighted Neighbor Joining algorithm to see if there are ways to reproduce the results, increase accuracy of branch lengths, and find similarities between the red panda mitochondrial DNA more easily. To confirm via another phylogenetic tree construction method, we implemented Parsimony phylogenetic methods utilizing dynamic programming. Utilizing the parsimony score, we are able to evaluate different potential trees to explain the phylogeny of the red panda, as well as explore neighboring trees for possible improvements. We compare both of the phylogenetic results with the neighbor joining algorithm as well as the phylogenetic tree from Joshi et al. [1] created using Bayesian methods. The more information we have surrounding the divergences of the two species within Red Pandas, the more sure we are about confirming this divergence in species as outlined in Joshi et al. [1].

Keywords (minimum 5): red pandas, phylogenetic tree, endangered species, divergence, monotypic taxon, Neighbor Joining, Parsimony, Weighted Neighbor Joining

Project type: Reimplementation

Project repository: https://github.coecis.cornell.edu/ars422/cs4775_final_project_red_pandas
https://github.com/ars422/cs4775_final_project_phylogeny_red_pandas

The Cornell repo is the one we worked out of, but if it is not accessible, you can utilize the second link.

1 Introduction

Phylogenetic methods are used to evaluate the evolutionary history of some group of species. The evolutionary history of the Red Panda is interesting to explore as it has been found to be a monotypic taxon, though historically, it was placed in the raccoon family and then the bear family. Analyzing this evolutionary history can provide meaningful insight on necessary conservation methods for the species. As explained in Joshi et al. [1], the Red Panda has been classified as an endangered species under the IUCN Red List. Human intervention, including anthropogenic activities and land fragmentation, have destroyed important habitats for the Red Panda. In addition, poaching has contributed to the endangerment of the species.

Joshi et al. [1] used Bayesian phylogenetic methods to better classify species boundaries for Red Pandas. Results showed the separation of the Red Panda into two species: the Himalayan Red Panda *Ailurus fulgens* and the Chinese Red Panda *Ailurus styani*. These boundaries are important in ensuring conservation methods can be better specialized for each species and therefore be more effective. Specifically, Joshi et al. [1] found the Siang river to be a geographic boundary in the Red Panda's evolutionary divergence, which can guide efforts to conserve specific regions of habitats. To support the importance of such findings regarding species conservation, we apply several phylogenetic methods different from those used in Joshi et al. [1] to confirm their results, including Neighbor-Joining, Weighted Neighbor-Joining, and Parsimony methods.

2 Methods

2.1 Dataset Preparation

We analyzed 53 red panda species haplotypes obtained from NCBI Genbank to ensure the sequences were in a usable format for downstream phylogenetic analysis (see Table 1 for sequences used). Sequence alignment was performed using two complementary tools: BLAST and ClustalW, each employing distinct alignment strategies. Initially, we utilized BLAST, which performs pairwise alignments between a query sequence and target sequences. The haplotype H1 was selected as the query sequence, and the following parameters were applied: gap open penalty = 5, gap extension penalty = 2, match score = 2, and mismatch penalty = -3. Subsequently, to validate these alignments and align more closely with the methodology of our base paper Joshi et al. [1], we reanalyzed the sequences using ClustalW. Unlike BLAST, ClustalW performs multiple sequence alignments, allowing for a global assessment of sequence homology. The parameters for ClustalW were set as follows: gap open penalty = 15, gap extension penalty = 6.66, match score = 1.9, and mismatch score = 0. In both cases, the sequences were aligned to have a length of 436 bp.

Special consideration was given to sequences containing ambiguous bases ("N"), which were treated as matches since "N" represents any of the four nucleotides (A, T, G, or C) as established by the International Union of Pure and Applied Chemistry (IUPAC) [2]. The choice of alignment tools and parameter settings was critical in ensuring consistency across the data, as alignment quality directly influences the accuracy of subsequent phylogenetic analyses. The pairwise alignments provided by BLAST enabled an initial identification of homologous sequences, while the global alignment from ClustalW ensured that the sequences were optimally aligned across their entire lengths for downstream analysis. By employing these complementary alignment strategies, we aimed to minimize biases and optimize sequence quality for constructing phylogenetic trees using Weighted Neighbor-Joining, Neighbor-Joining, and Parsimony methods. The alignment utilizing ClustalW was used in our analysis below.

2.2 Distance Matrix Calculation

The multiple sequence alignments generated from ClustalW were used to construct a distance matrix, which served as the input for the Weighted Neighbor-Joining algorithm and Neighbor-Joining method. This distance matrix provided the pairwise genetic distances required to infer phylogenetic relationships among the 53 red panda haplotypes.

To compute the distance matrix, first the Hamming distance matrix was calculated as a measure of the proportion of different base pairs between two aligned nucleotide sequences [3]. In this approach, each mismatch contributed one to the distance. The total number of mismatches was then divided by the sequence length, which in this analysis was 436 base pairs.

The resulting Hamming distances were used as inputs to the Jukes-Cantor model to estimate the evolutionary distances between nucleotide sequences. The Hamming distance between sequences can be misleading, and Jukes-Cantor correction method addresses this by accounting for multiple substitutions, providing a more accurate representation of evolutionary distance [4]. The Jukes-Cantor model assumes that all nucleotide substitutions occur at equal rates and provides a correction for multiple substitutions that may occur at the same site over time. The model calculates the evolutionary distance between two sequences, D, using the following formula:

$$D = -\frac{3}{4}\ln\left(1 - \frac{4}{3}p\right), \tag{1}$$

where p represents the proportion of differing sites (Hamming distance).

This combined approach of using Hamming distances and the Jukes-Cantor model allowed us to construct an accurate distance matrix as input for phylogenetic analyses using the Weighted Neighbor-Joining and Neighbor-Joining algorithms.

2.3 Neighbor-Joining

The Neighbor-Joining (NJ) method is a widely used distance-based algorithm for constructing phylogenetic trees, particularly from molecular sequence data. Introduced by Saitou and Nei [5], it provides an efficient means of estimating evolutionary relationships among a set of species or taxa based on their pairwise genetic distances. The algorithm is a greedy heuristic for the balanced minimum evolution criterion and generates an unrooted tree [6].

The primary goal of the NJ algorithm is to minimize the total branch length of the resulting tree. It iteratively merges the most similar taxa by performing pairwise distance calculations. The process begins with constructing a distance matrix, where each entry represents the genetic distance between two sequences. As taxa are merged into internal nodes, the distance matrix is updated to treat these newly formed nodes as single units in subsequent calculations.

A central aspect of the NJ algorithm is identifying a pair of taxa that are both closely related to each other and relatively distant from other taxa. This is achieved by minimizing the $Q(i,j)$ metric, defined as [6]:

$$Q(i,j) = (n-2)d(i,j) - \sum_{k \notin X} d(i,k) - \sum_{k \notin X} d(j,k) \tag{2}$$

Here, i and j are the nodes being considered, n is the total number of taxa, and $d(i,j)$ is the genetic distance between nodes i and j. Once the pair (i,j) that minimizes $Q(i,j)$ is identified, a new internal node z is introduced between them. The distances between z and the nodes i, j, and other taxa are calculated using the following equations [6]:

$$d(x,z) = \frac{1}{2}d(x,y) + \frac{1}{2(n-2)}\left[\sum_{k \notin X} d(x,k) - \sum_{k \notin X} d(y,k)\right] \tag{3}$$

$$d(y,z) = d(x,y) - d(x,z) \tag{4}$$

$$d(z,k) = \frac{1}{2}[d(x,k) + d(y,k) - d(x,y)], \quad \text{for } k \neq x,y \tag{5}$$

Here, x and y represent the nodes that minimize the Q matrix. After calculating the new distances, the current distance matrix D is updated to include the newly created node z and its respective distances to other nodes.

This iterative process continues until all taxa are merged into a single unrooted tree. The NJ algorithm's efficiency and simplicity make it a foundational tool in phylogenetics for analyzing evolutionary relationships.

2.4 Weighted Neighbor Joining

Weighted Neighbor-Joining (Weighbor) is a refined version of the traditional Neighbor-Joining (NJ) algorithm used to construct phylogenetic trees. This method enhances the accuracy and efficiency of evolutionary tree reconstruction by modeling distances as random variables, each with its own variance, and assuming these variables follow a Gaussian distribution [7]. The variance of these distances is calculated using the Jukes-Cantor distance matrix with the following equation [7]:

$$\sigma^2(d) = e^{\frac{8d}{3}} D(1-D)/L \tag{6}$$

where d is the distance, L is the sequence length, and D is the Hamming distance.

To join taxa efficiently, Weighbor evaluates two key criteria for each pair of neighbors: Additivity and Positivity. The additivity criterion ensures that the distances between taxa are consistent with the tree's structure, meaning the distances between adjacent nodes should be additive (i.e., they must be explainable by the branch lengths of the tree). The additivity score for a given pair of taxa i and j is computed using the following formula [7]:

$$\mathrm{Add}(i,j) \equiv \frac{1}{2}\sum_{k \notin \{i,j\}} \frac{\left[d_{ik} - d_{jk} - (d_{iP} - d_{jP})\right]^2}{\sigma^2_{\mathrm{nonadd}}(d_{iP}, d_{Pk}) + \sigma^2_{\mathrm{nonadd}}(d_{jP}, d_{Pk})}. \tag{7}$$

Here, d_{iP} is calculated using the following formula [7]:

$$d_{iP} = \frac{d_{ik} + d_{ij} - d_{jk}}{2} \tag{8}$$

For the additivity score to be computed, our Weighbor algorithm uses two helper functions. The first computes d_{Pk} using the following equation [7]:

$$d_{Pk} = \frac{(d_{ik} - d_{iP})/\sigma^2_{\text{avg}}(iP) + (d_{jk} - d_{jP})/\sigma^2_{\text{avg}}(jP)}{1/\sigma^2_{\text{avg}}(iP) + 1/\sigma^2_{\text{avg}}(jP)}. \tag{9}$$

The second computes the non-additivity variance, $\sigma^2_{\text{nonadd}}(d_{iP}, d_{Pk})$, using the following equation [7]:

$$\sigma^2_{\text{nonadd}}(d_{iP}, d_{Pk}) = \sigma^2(d_{ik}) - \sigma^2(d_{iP}) - \sigma^2(d_{Pk}) \tag{10}$$

The positivity criterion validates that internal branch lengths do not become negative, which would indicate an impossible evolutionary scenario. This involves calculating a positivity score that checks the likelihood of a branch length being positive, with penalties for negative lengths. Positivity can be evaluated using the following equation [7]:

$$\text{Pos}(i,j) \equiv -\ln\left(\frac{1}{2}\,\text{erfc}\left(\frac{-d_{PQ}}{\sqrt{2}\,\sigma_{PQ}}\right)\right). \tag{11}$$

Weighbor applies a heuristic approach to reduce computational complexity without sacrificing accuracy. At each iteration, for each remaining node i, the algorithm identifies the most likely sister node j based on the additivity and positivity scores, ultimately guiding the tree-building process.

2.5 Rooting tres

Both traditional Neighbor Joining (NJ) and Weighted Neighbor-Joining (Weighbor) produce unrooted trees; therefore, a method was required to root the trees. Rooting a tree is necessary to demonstrate the evolutionary pathways of biological species because an unrooted tree only represents the relationships among taxa without indicating the direction of divergence [8].

For both the Weighbor and NJ trees, a midpoint rooting method was employed. This approach calculates the midpoint of the longest path through the unrooted tree, ensuring a balanced rooting without assigning a specific species as an outgroup. Midpoint rooting provides a symmetrical representation of evolutionary relationships and avoids potential biases associated with outgroup selection, which may be particularly challenging when the evolutionary position of the outgroup is uncertain or ambiguous [8]. Using the phylo package in Biopython, we used a midpoint rooting function to root the trees and produce the visualizations in Figure 1 and Figure 2.

Using midpoint rooting ensures that the inferred evolutionary relationships in both the NJ and Weighbor trees are biologically plausible and unbiased by assumptions about which species diverged earliest. This method is especially appropriate for datasets such as the Red Panda's, where understanding divergence patterns without predefined outgroups can yield insights into the evolutionary history and relationships within the species.

2.6 Parsimony

Our main goal with parsimony was to implement a new method of evaluating phylogenetic trees. As finding the maximum parsimony tree is NP-Hard, we stuck to evaluating given trees, including the tree from Joshi et al. [1] generated from Bayesian methods, the tree produced by Weighted Neighbor Joining, and the tree produced by Neighbor Joining.

Parsimony leverages the idea of Occam's Razor, the notion that the simplest explanation is probably the correct one. It evaluates trees based on how many evolutionary changes would occur to produce the resultant tree. In comparing parsimony scores, the tree with the lowest parsimony score is the most parsimonious, or most likely according to parsimony.

We will be utilizing Sankoff's algorithm which leverages dynamic programming as a method for evaluating the smallest number of weighted state changes needed for a given tree's representation of the data. The total computation time of this algorithm is $O(mnk^2)$ where n is the number of samples, k is the number of character states, and m is the length of the sequence [9]. We have 53 mitochondrial DNA sequences from red pandas, aligned using ClustalW allowing for gaps. Therefore, the characters for our parsimony algorithm are $\{A, C, G, T, -\}$. We are working with $n = 53$ samples, $k = |\{A, C, G, T, -\}| = 5$ character states, and sequence length $m = 436$ base pairs.

Sankoff's algorithm is made up of two stages. The first stage works from the leaf nodes up to the root node to compute the optimal score at the root of the tree. The second state is a backtrace which finds the optimal internal node character assignments utilizing backpointers.

The base case for Sankoff's algorithm is applied to the leaf/tip nodes where t is a tip node $s_t(i) = 0$, if node t has state i and $s_t(i) = \infty$ otherwise [10].

The recurrence relation for Sankoff's algorithm is [10] [9]:

$$s_u(i) = min_j\{c_{ij} + s_l(j))\} + min_k\{c_{ik} + s_r(k)\}$$

such that $s_u(i)$ is the minimum parsimony score of the subtree rooted at node u, given that node u is in state i [10]. We calculate this score for all possible states i the node u can have. Essentially, the recurrence relation determines the state for each child of the parent node u that minimizes the parsimony score when node u is in state i. We calculate the score for u by taking the accumulated parsimony score of a child node for some state j and adding the transition cost from state j to the parent's state i, then adding these sums together over all children. After considering the possible states each child can have, we determine the minimum score for the parent. In our algorithm, the transition cost is one if the state changes between nodes, and zero otherwise.

We then add up all of the minimum parsimony scores from each index of the sequence to reach our final parsimony score of the entire sequence. It is worth noting that we assume independence between each index in the sequence.

However, this is only how we are calculating the parsimony score. As a method for searching for trees which have lower parsimony scores, we also implemented a hill-climbing algorithm to explore the neighborhoods of the given tree.

This hill-climbing algorithm works by first evaluating the score of the given tree. Then, it finds all of the neighbors using Nearest Neighbor Interchange (NNI), evaluating each of their parsimony scores. If a neighbor is found to have a lower score than the original, then we now repeat our process looking at the neighborhood of that tree and storing that tree as our optimal [11]. This occurs until we no longer have neighbors to be explored as our given tree has the lowest parsimony score found.

NNI operates by recursively finding each tree through interchanges of internal nodes [12]. These occur by recursively looking at all of our internal nodes and finding their sibling, a node they share a parent with. Then, two new trees are then created by first swapping the left subtrees of the two siblings to get one and then swapping the right subtrees of the two siblings to get another possible tree. It's important to note that our implementation of NNI does not allow for interchanges between siblings which include an internal node and a leaf node.

3 Results

3.1 Tree Comparison

The phylogenetic trees generated using the Weighbor algorithm (Figure 2) and Neighbor Joining (NJ) (Figure 1) revealed significant discrepancies in their topologies. These differences were apparent not only between the Weighbor and NJ trees but also in comparison to the reference tree published in the base paper by Joshi et al. [1]. Discrepancies were particularly pronounced in sections where taxa were closely related, leading to differences in branching patterns and the clustering of species.

The NJ tree exhibited a topology consistent with classical hierarchical clustering approaches, relying solely on pairwise distances without applying additional weighting. This method resulted in a more balanced topology but sometimes failed to capture subtle evolutionary dynamics among closely related species. On the other hand, the Weighbor tree, which incorporates taxon-specific weighting, substantially modified branch lengths and redefined clades. This led to an asymmetric topology and more pronounced delineation among taxa with highly divergent evolutionary rates.

Interestingly, neither NJ nor Weighbor produced a tree identical to the reference tree (see Figure 6, constructed using IcyTree as there were no branch lengths), which was constructed using Bayesian methods. The Bayesian approach likely incorporated site-specific evolutionary models and probabilistic inference, accounting for aspects of sequence evolution that distance-based methods inherently overlook. This divergence highlights how methodological differences, including data handling and model selection, shape phylogenetic outcomes.

Despite their differences, both the NJ and Weighbor trees aligned with Joshi et al. [1] in identifying two distinct clades within the red panda species. These clades, robustly supported across methods, reflect evolutionary divergence within the species and offer a point of validation for the generated trees.

The placement of closely related taxa varied significantly between the methods. In the NJ tree, taxa that were nearly indistinguishable in terms of pairwise distances clustered tightly, often forming unresolved polytomies. In contrast, the Weighbor tree, influenced by weighting, resolved some of these polytomies but occasionally misplaced taxa into unexpected groupings. This suggests that while weights can help resolve ambiguities, they may also introduce biases if taxon-specific weights are not optimally calibrated.

3.2 Parsimony Scores

We can compare the performance of each algorithm by calculating parsimony scores for each algorithm's resulting phylogenetic tree. We ran our implemented version of Sankoff's algorithm on each tree to determine the most parsimonious. Our results found the tree from Joshi et al. [1] created using Bayesian methods to have a score of 130, the tree created using NJ to have a score of 131, and the tree created using Weighbor to have a score of 307 (Figure 3). We consider the tree with the lowest score to be the most parsimonious meaning it has the fewest number of evolutionary changes. From these results, we see that our base tree is the most parsimonious, followed by the tree created using NJ, then the tree created using Weighbor.

Our base tree and the tree created using NJ resulted in very close parsimony scores with a difference of only one. This is likely due to the rooting method used in our NJ implementation. Our NJ algorithm first returns an unrooted tree to which we apply the midpoint rooting method. This adds an extra node which our base tree would not contain. Therefore, this extra node in the NJ tree results in additional transitions from its children that our base tree does not include. These transitions could add additional costs to the parsimony score which could explain the slight increase in the NJ tree's score from our base tree's score.

Weighbor resulted in a higher parsimony score of 307 likely because Weighbor attempts to hone in on the accuracy of the branch lengths and parsimony does not really take those into account when evaluating the tree. Parsimony only evaluates based on the parent-child relationship of nodes present in the tree.

Considering our NJ algorithm resulted in a close parsimony score to our base tree, we can reason that their interpretation of evolutionary changes could be quite similar. Since their scores were both relatively low compared with other trees found, we also find that this reaffirms the conclusion reached in Joshi et al. [1].

3.3 Comparing Runtimes

The execution times of the Weighted Neighbor-Joining (Weighbor) and regular Neighbor-Joining (NJ) algorithm highlights significant differences in performance. The Weighbor algorithm recorded an execution time of 6.89 seconds, whereas the regular NJ algorithm completed in just 0.03 seconds (see Figure 4). This difference arises from fundamental differences in their computation processes.

Weighbor incorporates advanced statistical modeling to improve tree accuracy, including likelihood-based scoring, variance calculations, and corrections for errors in distance estimates. These features make it robust against issues like long-branch attraction but add significant computational overhead. In contrast, the NJ algorithm prioritizes speed by using simpler distance-based criterion and avoids complex statistical evaluations, allowing it to run much faster.

We also see that Parsimony runs much faster than Weighbor but still slower than NJ. It still has to process and run on every internal node for every index in the sequence, through a two-step dynamic program. On the other hand, NJ is a greedy algorithm with heuristics for choice as it goes along, so it is able to run faster.

The tradeoff between speed and accuracy is evident in these results. While NJ is well-suited for large datasets where execution time is critical, Weighbor ideally offers more reliable results for datasets with significant variability or noise. By modeling variances and focusing on statistical consistency, Weighbor should deliver higher accuracy at the cost of increased running time. Though in our case our parsimony scores were higher for Weighbor, with more information inputted and taken into account, we hope it will reach greater accuracy.

3.4 Utilizing NNI

Finally, since our Weighbor algorithm found the highest parsimony score out of the methods, we utilized β-Hill Climbing as a method for finding a better tree jumping off of the generated tree. When it found a neighboring tree with a lower parsimony score, it looked to that tree's neighbors for other options. The tree that was found (see Figure 5, constructed using IcyTree as there were no branch lengths outputted) had a parsimony score of 296, which was lower than the original Weighbor tree with score 307. It's important to note that the resultant tree was not the only neighboring tree that had a parsimony score of 296. The algorithm prioritizes the first tree found.

The NNI-Hill-Climbing Algorithm was also run on both the tree from Joshi et al. [1] and the resultant tree from NJ, but both did not yield significant results.

We can see that the resultant tree (Figure 5) still had an initial division which implies a division in clades that is present in the tree from Joshi et al. [1]. Though, it's important to note that the trees (Figures 6 and 5) themselves are a bit flipped in terms of ordering of leaf nodes (top versus bottom).

4 Discussion

The phylogenetic analysis of Red Pandas conducted in this study corroborates the findings of Joshi et al. [1] and provides additional insights into the evolutionary divergence of Himalayan Red Pandas *Ailurus fulgens* and the Chinese Red Panda *Ailurus styani*. By employing Weighted Neighbor-Joining and Parsimony methods, we reaffirm the Siang River's role as a geographic boundary and suggest potential interest in catering conservation strategies to this knowledge. For a greater breakdown of locations of each haplotype, refer to Joshi et al. [1] which contains a map of each surrounding the Siang River.

Our study successfully validated the separation of Red Panda haplotypes into two major clades, corresponding to the Himalayan Red Pandas *Ailurus fulgens* and the Chinese Red Panda *Ailurus styani* lineages. The use of Weighted Neighbor-Joining provided refined branch lengths and an outcome which most likely aligns with the divergence articulated in the tree generated using Bayesian phylogenetic methods, while parsimony methods reinforced these findings through independent scoring. Additionally, the incorporation of the Jukes-Cantor model for evolutionary distance calculations improved the reliability of our phylogenetic tree reconstructions. The inclusion of novel haplotypes from under-sampled regions by Joshi et al. [1] added robustness to our data and further substantiated the Siang River's role as a natural evolutionary barrier.

One of the primary strengths of this study is the methodological diversity, which allowed for cross-validation of results through multiple phylogenetic approaches. By utilizing both Weighted Neighbor-Joining and Parsimony methods, the study reduced potential biases associated with single-method analyses. Additionally, the use of many different available haplotypes expanded the geographic and genetic scope of the dataset, ensuring a more comprehensive coverage of the Red Panda's evolutionary range. The consistent alignment of our findings with prior studies (e.g., Joshi et al. [1]) further emphasizes the reliability of our conclusions.

Despite its strengths, the study has limitations. The reliance on mitochondrial DNA sequences, while effective for phylogenetic reconstructions, may not fully capture the genomic diversity of red pandas. This limitation could result in incomplete insights into gene flow amongst descendants and adaptive divergence. Additionally, the dataset, while covering many locations, still lacks representation from certain regions, such as Nepal, Bhutan, and the eastern extent of Chinese Red Panda's range. These gaps could influence the comprehensiveness of our findings.

To address these limitations, future research could prioritize the inclusion of nuclear genomic data, RNA, or regular DNA, which could offer a more detailed perspective on genetic diversity and evolutionary dynamics. Expanding the geographic scope to include underrepresented regions would further strengthen the phylogenetic framework. Additionally, integrating ecological and behavioral data could provide context for observed genetic patterns, aiding in brainstorming more species-specific conservation strategies. Advanced phylogenetic methods could also refine the divergence time estimates and shed light on the historical factors influencing Red Panda evolution. In addition, if we wanted to further explore the Siang river as the geological boundary for speciation, we could utilize similar phylogenetic methods on other species in the region such as insects to see if it impacts them as well.

Overall, this study has enhanced our understanding of the divergence of the Red Panda species, and it highlights the critical role of geographic barriers in shaping evolutionary trajectories. By analyzing the potential genetic history of red pandas, we are able to deepen our understanding of them.

References

1. Joshi BD, Dalui S, Singh SK, Mukherjee T, Chandra K, Sharma LK, Thakur M, 2020. New insights on species divergence in red panda. *bioRxiv*, .

2. IUPAC-IUB Commission on Biochemical Nomenclature, 1974. Abbreviations and symbols for nucleic acids, polynucleotides and their constituents. *Pure and Applied Chemistry*, 40(3):277–331.

3. Bookstein A, Kulyukin VA, Raita T, 2020. Generalized hamming distance. *Information Retrieval*, 5:353–375.

4. Gabriel EF, Jeff UI, 2024. Jukes-cantor correction for phylogenetic tree reconstruction. *bioRxiv*, .

5. Saitou N, Nei M, 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425.

6. Kim J, 2024. Lecture 12: Phylogenetics: Distance methods. Canvas, Course CS 4775.

7. Bruno WJ, Socci ND, Halpern AL, 2000. Weighted neighbor joining: A likelihood-based approach to distance-based phylogeny reconstruction. *Molecular Biology and Evolution*, 17(1):189–197.

8. Kinene T, Wainaina J, S Maina LB, 2016. Rooting trees, methods for. *Encyclopedia of Evolutionary Biology*, :489–493.

9. Kim J, 2024. Lecture 16: Phylogenetics: Parsimony methods. Canvas, Course CS 4775.

10. Kannan L, Wheeler WC, 2014. Exactly computing the parsimony scores on phylogenetic networks using dynamic programming. *Journal of computational biology : a journal of computational molecular cell biology*, 21(4):303–319.

11. Al-Betar MA, 2017. β-hill climbing: an exploratory local search. *Neural Computing and Applications*, 28(1):153–168.

12. Fischer M, 2024. On the correctness of maximum parsimony for data with few substitutions in the NNI neighborhood of phylogenetic trees. *Annals of Combinatorics*, .

Figures

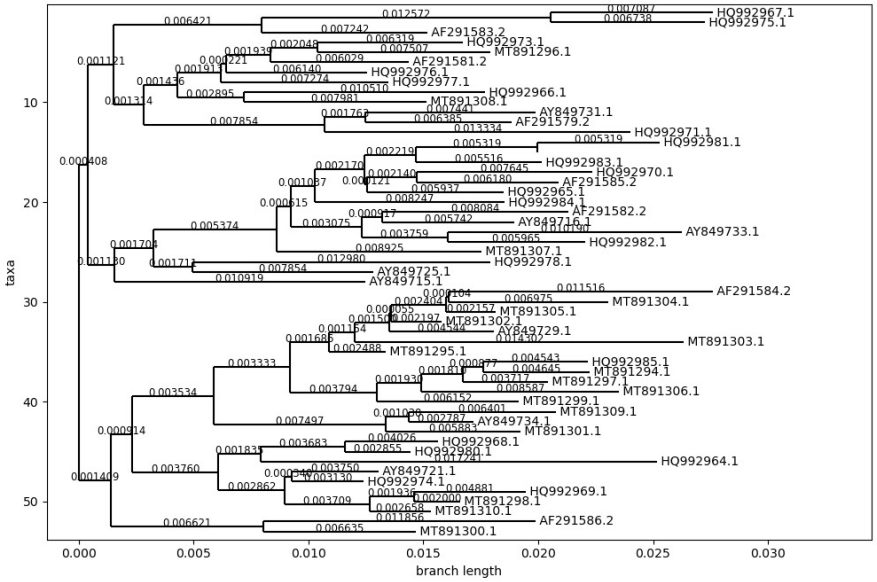

Figure 1. **NJ** Output tree from Neighbor-Joining generated using BioPython

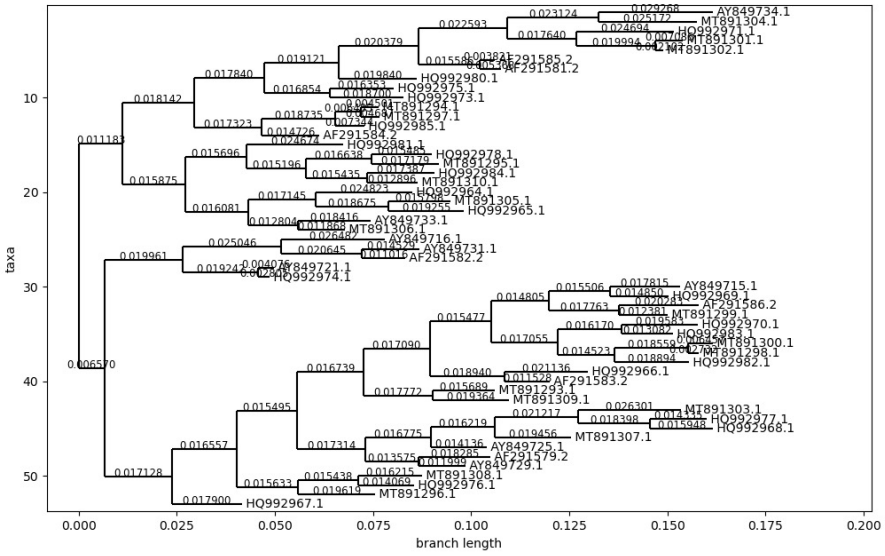

Figure 2. **WNJ** Output tree from Weighbor generated using BioPython

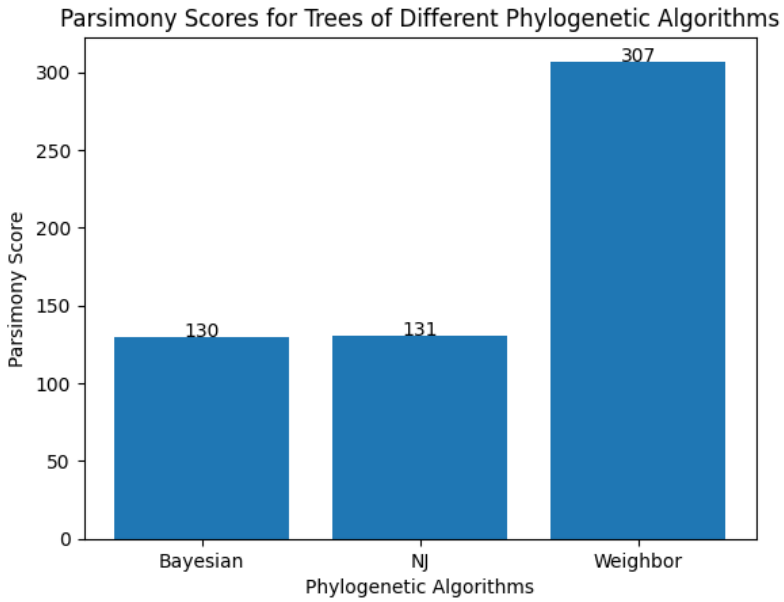

Figure 3. Parsimony Scores Evaluating Differing Tree Possibilities

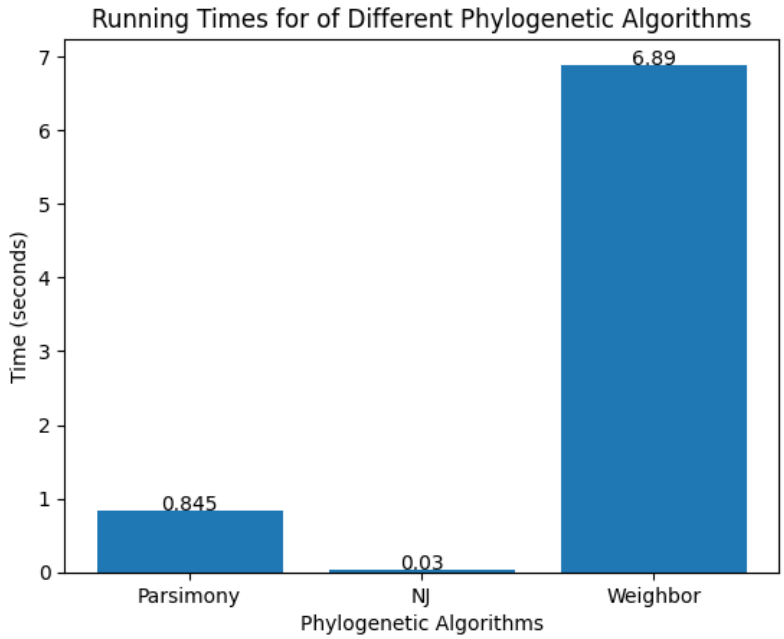

Figure 4. Runtimes Evaluating Differing Tree Algorithm Runtimes

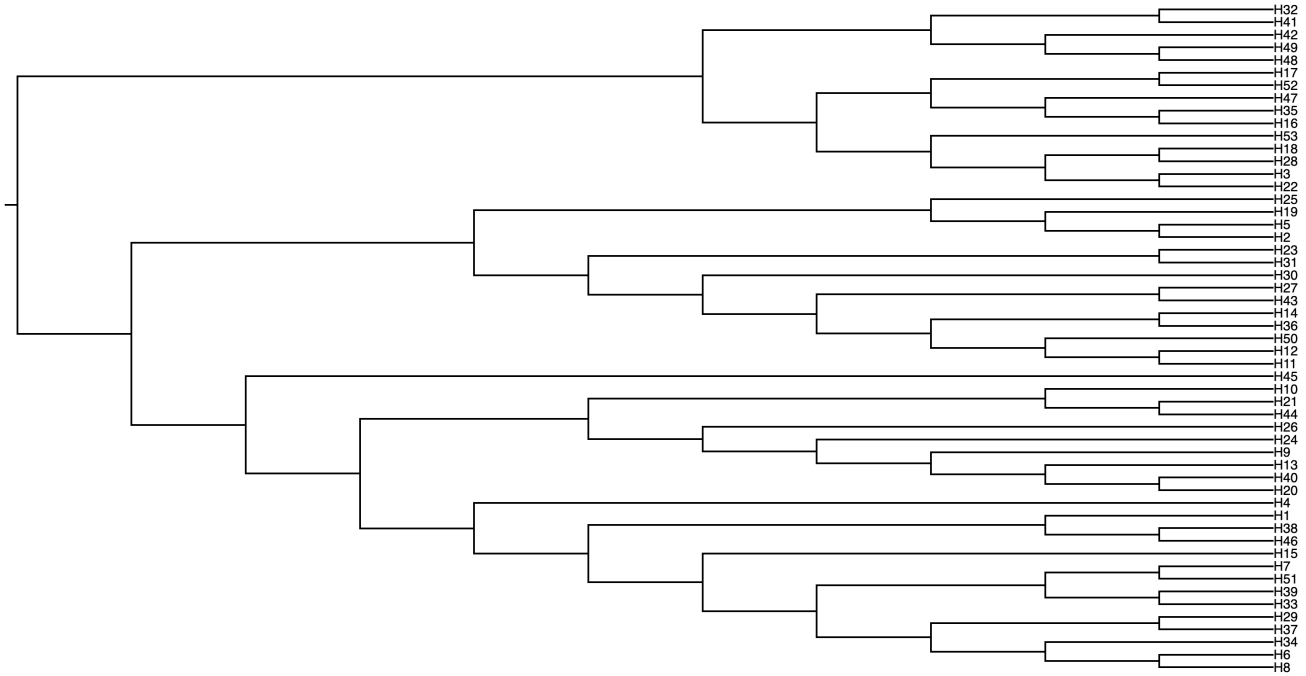

Figure 5. NNI Tree Tree From Utilizing β -Hill Climbing on Weighbor Tree using IcyTree

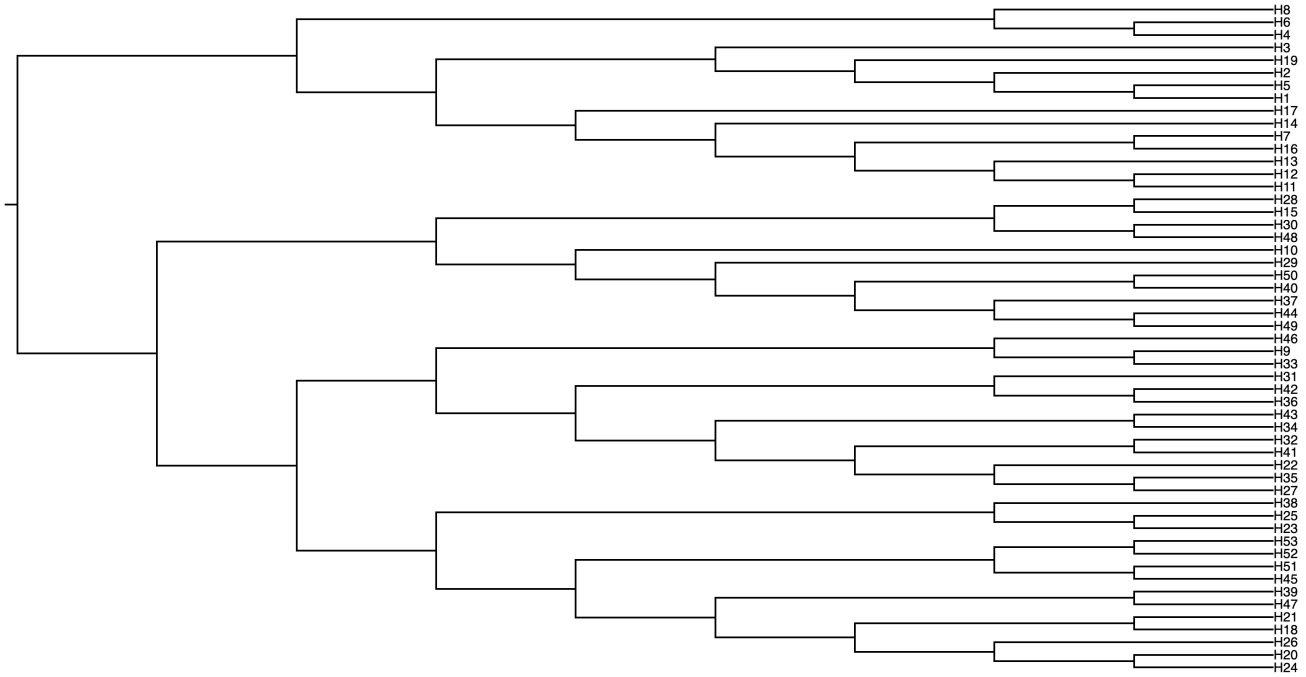

Figure 6. Tree from Joshi et al. [1] Original Tree Displayed using IcyTree

Haplotype	Accession Number	Locality	State	Country
H1	MT891293	Darjeeling zoo	West Bengal	India
H2	MT891294	West Sikkim/Singalia/East Sikkim	Sikkim/West Bengal/Sikkim	India
H3	MT891295	West Sikkim, East Sikkim	Sikkim	India
H4	MT891296	West Sikkim/Neora Valley NP	Sikkim / West Bengal	India
H5	MT891297	Neora Valley NP /East Sikkim	West Bengal / Sikkim	India
H6	MT891298	Neora Valley NP	West Bengal	India
H7	MT891299	Tawang / Mandala / Zingchang	Arunachal Pradesh	India
H8	MT891300	West Sikkim / East Sikkim / Neora Valley NP	Sikkim / West Bengal	India
H9	MT891307	Anini, Dibang	Arunachal Pradesh	India
H10	MT891308	Zingchang, Mandala	Arunachal Pradesh	India
H11	MT891301	Zingchang, Mandala	Arunachal Pradesh	India
H12	MT891302	Mandala	Arunachal Pradesh	India
H13	MT891303	Mandala	Arunachal Pradesh	India
H14	MT891304	Mandala	Arunachal Pradesh	India
H15	MT891309	Lower Dibang	Arunachal Pradesh	India
H16	MT891305	Mechuka, West Siang	Arunachal Pradesh	India
H17	MT891306	Khellong	Arunachal Pradesh	India
H18	MT891310	Anini, Dibang	Arunachal Pradesh	India
H19	HQ992985.1	Gaoligong	Western Yunnan / North Myanmar	China
H20	HQ992977.1	Gaoligong	Western Yunnan / North Myanmar	China
H21	AY849729.1	South East Tibet	Tibet	China
H22	HQ992978.1	Gaoligong	Western Yunnan / North Myanmar	China
H23	HQ992973.1	Qionglai	North Sichuan	China
H24	AY849725.1	Gaoligong	Western Yunnan/North Myanmar	China
H25	AF291584.2	Qionglai	North Sichuan	China
H26	HQ992976.1	Gaoligong	Western Yunnan/North Myanmar	China
H27	AF291581.2	Qionglai	North Sichuan	China
H28	HQ992984.1	South Eastern Tibet	Tibet	China
H29	HQ992982.1	Gaoligong	Western Yunnan/North Myanmar	China
H30	HQ992980.1	Gaoligong	Western Yunnan/North Myanmar	China
H31	HQ992975.1	Gaoligong	Western Yunnan/North Myanmar	China
H32	HQ992974.1	Xiaoxiangling	South central Sichuan	China
H33	HQ992970.1	Liangshan	South Sichuan	China
H34	HQ992969.1	Xiaoxiangling	South central Sichuan	China
H35	HQ992965.1	Liangshan	South Sichuan	China
H36	AY849734.1	Gaoligong	Western Yunnan/North Myanmar	China
H37	AY849715.1	Xiaoxiangling	South central Sichuan	China
H38	AF291583.2	Qionglai	North Sichuan	China
H39	HQ992983.1	Gaoligong	Western Yunnan/North Myanmar	China
H40	HQ992968.1	Liangshan	South Sichuan	China
H41	AY849721.1	Liangshan	South Sichuan	China
H42	AY849716.1	Gaoligong	South Yunnan	China
H43	AF291585.2	Qionglai	North Sichuan	China
H44	AF291579.2	Liangshan	South Sichuan	China
H45	HQ992967.1	Qionglai	North Sichuan	China
H46	HQ992966.1	Qionglai	North Sichuan	China
H47	AY849733.1	Gaoligong	Western Yunnan/North Myanmar	China
H48	AY849731.1	Gaoligong	Western Yunnan/North Myanmar	China
H49	AF291582.2	Liangshan	South Sichuan	China
H50	HQ992971.1	Liangshan	South Sichuan	China
H51	AF291586.2	Qionglai	North Sichuan	China
H52	HQ992964.1	Qionglai	North Sichuan	China
H53	HQ992981.1	Gaoligong	Western Yunnan/North Myanmar	China

Table 1. Table of Haplotypes, GenBank Accession Numbers, and Locations Utilized [1]