Supplementary Method S1: Detailed Description of the VisualHyDe Script and a Recommended Workflow for Analyzing HyDe Results

Introduction

The detection and interpretation of hybridization and introgression events from genomic data are crucial for understanding evolutionary processes. The software HyDe provides a robust statistical framework for identifying hybridization at the level of individual taxa by analyzing phylogenetic quartet frequencies. However, interpreting the extensive output of HyDe across a large phylogeny, and distinguishing genuine signals from statistical artifacts, can be challenging.

To address these challenges, we developed VisualHyDe, a novel Python script designed to visualize, analyze, and critically evaluate hybridization signals. This document is divided into two main parts. **Part I** provides a detailed technical description of the script, including its input requirements, the algorithms for its **Leaf** and **Node** analysis modes, and its false-positive filtering system. **Part II** outlines a comprehensive, step-by-step recommended workflow for applying VisualHyDe, from initial data simulation to the final, robust visualization of hybridization events.

Part I: Description of the VisualHyDe Script

1. Input Data Requirements

The VisualHyDe script requires specific input files to function correctly. Strict consistency in taxon nomenclature across all files is mandatory; any mismatch will result in a fatal error to prevent erroneous analysis.

- **Species Tree:** A fully resolved species tree in Newick format (-t or --treefile). This tree provides the topological framework for all analyses. The tree must be rooted with a single outgroup. Furthermore, if the false positive filtering workflow is to be used, this should be a maximum likelihood tree with accurate branch lengths.
- **HyDe Output File:** A tab-delimited file generated by HyDe (-i or --infile). The script requires a header and the following five columns: P1, P2, Hybrid, Gamma, and Pvalue. The Pvalue column is used to filter for statistically significant hybridization triplets.
- **Optional Predefined Clade File:** A comma-separated text file (-c or --preclade) that defines specific clades for color-highlighting on the tree visualization. Each line represents a clade, with taxon names separated by commas (e.g., SpeciesA,SpeciesB,SpeciesC,). If not provided, the script will automatically generate a simple clade definition file.
- **Optional Null Hypothesis Filter Directory:** A directory containing null-hypothesis data for false-positive filtering (--filter_dir). This directory must contain a set of .csv files, each corresponding to a specific leaf or node analyzed. The naming convention must strictly match the output names generated by the script (e.g., SpeciesA.csv for a leaf, Node_1.csv for an internal node). The internal structure (row and column names) of these CSV files

must also perfectly match the taxon names in the main species tree. For a detailed explanation of how these files are generated and used, see Part I, Section 4: False Positive Filtering via Null Hypothesis Simulation.

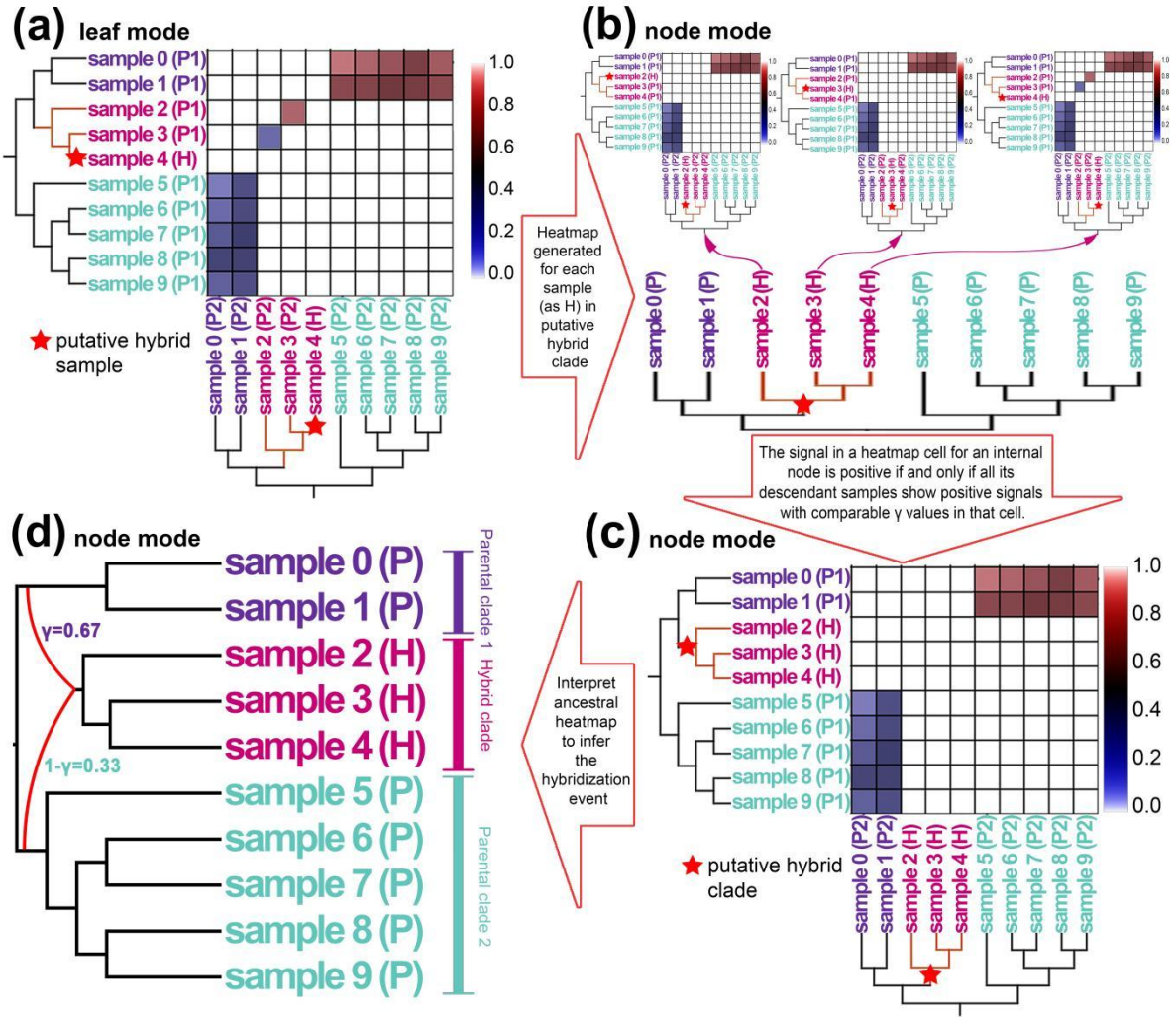

Figure. Schematic diagram illustrating hybridization detection and ancestral inference using HyDe and the VisualHyDe python script, which includes 'Leaf Mode' and 'Node Mode' functionalities. (a) Visualization from VisualHyDe's 'Leaf Mode' for an individual sample: A heatmap generated for a single putative hybrid (H = sample 4, marked with red star). The species tree annotates rows (potential P1) and columns (potential P2). Each colored square indicates a statistically significant HyDe test ((P1, H, P2), O), with color intensity corresponding to the estimated inheritance probability γ (proportion derived from row taxon P1). Blocks of similar colors can suggest the involvement of ancestral parental clades. (b) Hybridization signals in descendant samples: Individual Leaf Mode heatmaps for samples within a putative hybrid clade (samples 2, 3, 4, orange clade). These display the signals observed in descendant lineages, forming the basis for ancestral inference using Node Mode. (c)

Inferred ancestral heatmap using VisualHyDe's 'Node Mode': The heatmap estimating hybridization signals for the Most Recent Common Ancestor (MRCA) of the hybrid clade (samples 2, 3, 4). In this mode, a signal (colored cell) is inferred for the ancestor if and only if a "difference consensus" is achieved across its descendant lineages. This means that for a specific parental pair (P1, P2), a significant majority of the descendant internal nodes must show a highly consistent gamma signal between their respective children (i.e., a very low gamma difference). This robust test ensures that only ancestral signals validated by widespread and consistent concordance across descendants are retained. (d) Phylogenetic network interpretation: A schematic phylogenetic network derived from the inferred ancestral signals in (c) generated via Node Mode. The reticulation edge points to the MRCA of the hybrid clade, representing the inferred ancestral hybridization event. Associated values indicate estimated inheritance probabilities (γ=0.67 and $1-\gamma$=0.33) from the distinct ancestral parental lineages (Parental Clade 1 and another contributor)

2. Leaf Mode Analysis

The Leaf Mode is the fundamental analysis level, designed to visualize the hybridization profile for a single terminal taxon (a leaf).

For a given leaf *H*, the script first filters the HyDe output to retain only those triplets where *H* is the putative hybrid and the associated p-value is below a user-defined threshold (-p, default: 0.05). Subsequently, it constructs a square matrix where both rows and columns represent all potential parental taxa in the ingroup.

The value for each cell (P1, P2) in this matrix is the gamma statistic, which quantifies the proportion of the hybrid's genome inherited from parent P1. In the heatmap visualization, the P1 parent is represented on the y-axis (left side) and the P2 parent on the x-axis (bottom side). The gamma statistic from HyDe is directional; for a tested triplet (Parent A, Parent B, Hybrid), the gamma value quantifies the contribution from Parent B. To ensure the heatmap is intuitive, the script processes this information so that a cell's value always represents the contribution from the parent on the y-axis (P1). This is handled as follows:

- If HyDe reports a significant triplet as (P1, P2, H), its gamma value represents the contribution from P2. The script therefore calculates 1 - gamma to represent P1's contribution and assigns this value to the cell (row=P1, col=P2).
- If HyDe reports the triplet as (P2, P1, H), its gamma value represents the contribution from P1. The script assigns this gamma value directly to the cell (row=P1, col=P2).

In simpler terms, the color of a cell in the final heatmap represents the proportion of genetic material contributed by the parental taxon listed on the y-axis (left side). The resulting matrix is then visualized as a heatmap, providing an intuitive overview of the most likely parental pairs for the taxon *H* (Fig. a).

3. Node Mode Analysis: The Difference Consensus Algorithm

While analyzing hybridization on a leaf-by-leaf basis is informative, it may fail to capture ancient introgression events that occurred in the ancestor of a larger clade. A signal detected in a single taxon could be a recent, lineage-specific event or statistical noise. Conversely, a hybridization signal that is consistently detected across multiple descendant lineages provides much stronger evidence for a more ancient and impactful evolutionary event. The Node Mode is therefore an advanced feature designed to move beyond individual taxa and infer these ancestral hybridization signals at the internal nodes of the phylogeny. The general principle is to aggregate the potentially noisy or variable signals observed across multiple descendant lineages (Fig. b) to reconstruct a single, robust ancestral heatmap for their most recent common ancestor (Fig. c). This process effectively treats the descendant clade as a population of signals, seeking a consistent pattern that rises above the noise of individual terminal taxa. When a strong and consistent signal emerges from this aggregation, it can be interpreted with higher confidence as a specific reticulation event in the clade's deeper evolutionary history (Fig. d).

Inferring an ancestral state, however, presents a significant challenge. We posit that an ancestral hybridization event should produce signals with similar gamma values in its descendants, but determining the appropriate threshold and method to assess this similarity is a non-trivial problem. It is necessary to balance sensitivity to genuine, clade-wide signals with robustness against noise from terminal branches, which may arise from factors like poor sequence quality or lineage-specific evolutionary idiosyncrasies. A simplistic approach using a single, strict consistency threshold would be overly brittle; if the descendant lineages of just one terminal node fail this strict test, that node would register no signal. This would cause a cascading failure up the tree, potentially erasing a legitimate and widespread ancestral signal entirely.

To overcome this, we developed a novel two-stage "difference consensus" algorithm. This method is specifically designed to be resilient to localized noise while rigorously testing for clade-wide signal consistency. The core of this approach lies in the use of two distinct thresholds—a *strict* and a *loose* one—to evaluate signal consistency. The strict threshold is used to ensure that the vast majority of signals within a descendant clade are highly congruent. The loose threshold, in turn, serves as an upper bound for the few acceptable outliers, ensuring that even signals that do not meet the strict criterion are not wildly divergent. This two-tiered system allows the algorithm to be both rigorous in its demand for a strong consensus and resilient to minor, localized inconsistencies that might otherwise disrupt the analysis.

3.1 Stage 1: Postorder Traversal for Signal Aggregation

The purpose of this first stage is to act as an initial, broad filter, ensuring that any signal passed upwards for further consideration meets a baseline level of consistency, defined by the loose threshold. Signals that fail even this lenient check are discarded immediately. The algorithm begins with a postorder (bottom-up) traversal of the species tree. This ensures that when any internal node is processed, the results for its descendant children are already available. During

this stage, two preliminary data maps are computed for each internal node.

- **Draft Heatmap (draft_heatmaps)**: For each internal node N with direct children C1 and C2, and for each potential parental pair (P1, P2), the script retrieves the corresponding gamma-values from the draft heatmaps of its children, *gamma_C1* and *gamma_C2*. If the absolute difference |gamma_C1 - gamma_C2| is below a user-defined *loose* threshold (-gdt, default: 0.2), the average value (gamma_C1 + gamma_C2) / 2 is stored in the draft heatmap for node N. This loose threshold ensures that potential signals are aggregated and passed up the tree, preventing premature signal loss due to minor inconsistencies in terminal branches.
- **Difference Map (diff_heatmaps)**: Concurrently, for each cell (P1, P2), the absolute difference |gamma_C1 - gamma_C2| is calculated and stored in a separate "difference map" for node N. This data map, which is not visualized, serves to capture the degree of signal consistency between the two child lineages.

3.2 Stage 2: Preorder Traversal for Consensus Filtering

After the initial bottom-up pass, a second, preorder (top-down) traversal is performed to refine the results and generate the final heatmaps. The purpose of this second stage is to apply a much more stringent test: for a signal at an ancestral node to be considered robust, it must be demonstrated that the majority of its descendant lineages show a gamma difference below the strict threshold. For each internal node N and for each cell (P1, P2), a "difference consensus" test is conducted:

- **Data Collection**: The algorithm collects a list of all gamma difference values for the cell (P1, P2) from the diff_heatmaps of all internal nodes within the subtree rooted at N.
- **Consensus Test**: It then calculates the proportion of these collected difference values that are below a *strict* threshold (--diff_consensus_threshold, default: 0.1).
- **Decision**: If this proportion exceeds a user-defined consensus ratio (--diff_consensus_ratio, default: 0.8), it signifies a strong consensus of signal consistency throughout the entire descendant clade. In this case, the corresponding gamma-value from the draft_heatmap of node N is accepted and placed into the final heatmap. If the test fails, the signal is considered inconsistent and the cell is left empty (NaN).

This two-stage algorithm is robust against localized noise; a single inconsistent signal at a terminal branch will not cause the entire ancestral signal to be discarded, as long as the broader consensus of high consistency is met across the clade.

4. False Positive Filtering via Null Hypothesis Simulation

A key feature of VisualHyDe is its ability to perform a sophisticated false-positive correction, which is critical for robust hybridization inference. Methods based on phylogenetic quartet frequencies, such as the ABBA-BABA test underlying HyDe, were originally developed to detect introgression between very closely related taxa, such as different populations within the same species. In these original scenarios, the rates of molecular evolution among the compared lineages are expected to be highly similar. However, when these methods are applied to more

distantly related clades, significant variations in evolutionary rates between lineages can become a major confounding factor. A lineage that has evolved particularly fast can accumulate substitutions that coincidentally mimic the site patterns of gene flow, leading to statistically significant but biologically incorrect signals of hybridization.

To address this challenge, our approach is to model this source of error directly. By simulating sequence data under a "null hypothesis" of no hybridization—using the user-provided species tree topology and branch lengths—we can generate a dataset where any detected hybridization signal is, by definition, a false positive. These simulated results serve as a baseline for systemic bias inherent to the dataset's specific topology and evolutionary rates. By providing a directory of these pre-computed null-hypothesis results (--filter_dir), VisualHyDe can perform a background correction. The logic for applying this filter differs critically between the two modes to ensure the most appropriate correction is applied.

4.1 Generation of Null-Hypothesis Data

The generation of the required null-hypothesis filter files is a multi-step process that must be performed by the user prior to the final analysis. The process begins with the same species tree (e.g., a maximum likelihood tree) used for the primary analysis. Using a forward-time sequence evolution simulator, sequence alignments are simulated along the branches of this tree. It is critical that this simulation process adheres to the null hypothesis, meaning it should only model processes like mutation and drift, without any gene flow or hybridization. The resulting simulated alignments are then processed with HyDe in the same manner as the empirical data. Finally, the VisualHyDe script itself must be run on the HyDe output from the simulated data—once in Leaf Mode for all relevant taxa, and once in Node Mode for all internal nodes—to generate the complete set of .csv files that constitute the null-hypothesis filter directory.

4.2 Filtering Logic: Mode-Specific Application

The rationale for applying the filter differs between the two modes, reflecting their distinct analytical goals. The key is to apply the most relevant null-hypothesis baseline to each result without compromising the integrity of the ancestral inference algorithm.

- **Leaf Mode**: The filter is applied directly during the initial heatmap calculation. For each leaf, its corresponding null-hypothesis CSV is used to filter its real-data heatmap. This is appropriate as the false positives for a leaf are best modeled by its own null simulation.
- **Node Mode**: The filter is applied only as a **post-processing step**, *after* the entire two-stage consensus algorithm is complete for a given node. During the internal calculations of Node Mode, the leaf-level filters are deliberately ignored. This is crucial to ensure that the consensus algorithm operates on complete, unbiased input data from all descendant leaves. After the final heatmap for a node (e.g., Node_1) is computed, the script then uses the corresponding null-hypothesis file (Node_1.csv) to perform the filtering.

4.3 The Relative Strength Test

A simple filtering approach that removes any signal present in the null simulation would be overly punitive, as minor stochastic noise in the null data could erase a strong, genuine signal in the empirical data. To address this, the script employs a more nuanced 'relative strength test' to determine if a signal in the real data is significantly stronger than the background noise observed in the null simulation. For a given cell (P1, P2) with a real-data signal *gamma_real* and a null-simulation signal *gamma_null*:

- **Standardization**: Both gamma-values are standardized to a 0-0.5 scale to account for the symmetric nature of the statistic (a gamma-value of 0.1 is equivalent in strength to a gamma-value of 0.9). The standardized value, let's call it *s(gamma)*, is calculated as the minimum of the original gamma and (1 - gamma).
- **Ratio Calculation**: The ratio *R* is then calculated by dividing the standardized null signal by the standardized real signal: *R = s(gamma_null) / s(gamma_real)*.
- **Decision**: The real signal *gamma_real* is considered a potential false positive and is removed (set to NaN) if this ratio R is greater than or equal to a user-defined threshold (--filter_ratio_threshold, default: 0.05). This indicates that the signal observed in the null simulation is non-negligible in strength compared to the signal in the real data.

Part II: A Recommended Analysis Workflow

This section outlines a comprehensive workflow for utilizing VisualHyDe, from initial data preparation to the final filtered visualization, ensuring robust and reliable inference of hybridization.

Data Preparation

The quality of the input data is paramount. We recommend the following:

- **Sequence Alignment**: A concatenated, multi-locus sequence alignment is required. Based on our experience, an alignment length of at least 1 million base pairs (bp) is recommended to provide sufficient phylogenetic signal. The alignment must contain exactly one outgroup taxon, which is necessary for correct rooting and subsequent HyDe analysis. It is also crucial that the outgroup is not excessively divergent from the ingroup, as a very distant outgroup can lead to strong false positives in HyDe analysis.
- **Species Tree Inference**: A maximum likelihood (ML) species tree should be inferred from the alignment. We recommend using software such as RAxML. For this workflow, we will refer to the output of RAxML-HPC-PTHREADS. Two key output files are required:
    - The best-scoring ML tree (RAxML_bestTree.*), which must be correctly rooted using the designated outgroup.
    - The information file (RAxML_info.*), which contains the estimated substitution model parameters (e.g., GTR+G rates). These parameters are essential for realistic sequence simulation.

Step 1: Simulation of the Null-Hypothesis Dataset

The first step is to generate a sequence alignment under the null hypothesis of no hybridization. This is accomplished using forward-time sequence evolution simulation, which is implemented in the **pyvolve Python module**. To streamline this process, we used a wrapper script (e.g., AlignmentSimulator, available at https://github.com/Jhe1004/AlignmentSimulator/) to run pyvolve. This method simulates the evolution of nucleotide sequences along the fixed topology and branch lengths of the provided species tree. The script requires the rooted ML tree (RAxML_bestTree.*) and the substitution model parameters from the RAxML_info.* file as input. We recommend simulating a substantial alignment length, for instance, 5 million bp or more, to ensure that even weak false positive signals can be detected. The output of this step is a simulated sequence alignment that reflects the species tree's evolutionary history but is, by design, devoid of any gene flow.

Step 2: HyDe Analysis on Empirical and Simulated Data

Next, run the HyDe software separately on both the empirical (real) sequence alignment and the simulated null-hypothesis alignment. A critical consideration at this stage is the construction of the HyDe mapping file. To maximize the information retained from all individuals, each sample should be treated as an individual taxon and assigned its own unique ID in the mapping file, even if multiple samples belong to the same species or population. For both runs, it is important to use the unfiltered HyDe output file (the one containing all tested triplets, not just those passing a default significance filter), as VisualHyDe will perform its own filtering based on the provided p-value.

Step 3: Generation of Null-Hypothesis Filter Layers

With the HyDe results from the simulated data, the next step is to generate the filter files. Run the VisualHyDe script on this simulated HyDe output, using the original ML tree as the topological guide. This must be done twice:

- **Leaf Mode Run**: Run the script in its default Leaf Mode to generate a .csv file for every taxon.
- **Node Mode Run**: Run the script again on the same simulated data, but this time activating Node Mode with the -n flag. This will generate a .csv file for every internal node.

After these two runs are complete, collect all generated .csv files and move them into a single, newly created directory. This directory will serve as the input for the --filter_dir parameter in the final step.

Step 4: Final Analysis of Empirical Data with Filtering

Finally, you can analyze the empirical data. We strongly recommend a two-pass approach:

- **Initial Unfiltered Analysis (Optional)**: First, run VisualHyDe on the HyDe output from your empirical data *without* using the --filter_dir parameter. This will produce a raw visualization of all statistically significant signals. While informative, these results may

contain false positives influenced by branch length heterogeneity. It is advisable to move the output images and .csv files from this run into a separate backup folder to prevent them from being overwritten.

- **Filtered Analysis (Recommended)**: Re-run VisualHyDe on the same empirical HyDe output and species tree. This time, however, include the --filter_dir parameter, providing the path to the directory containing the null-hypothesis .csv files created in Step 3. The script will now perform the "relative strength test" for every signal, effectively removing false positives caused by systemic biases. The resulting visualizations are corrected for these biases and represent a more robust and reliable depiction of the hybridization landscape.

This workflow ensures that the final visualizations are not only comprehensive but also rigorously controlled for the most common sources of false positives in hybridization analyses.

Conclusion

The VisualHyDe script provides a powerful and flexible platform for the post-analysis of HyDe results. By integrating visualization with a novel "difference consensus" algorithm for ancestral state inference and a robust, mode-aware false-positive filtering mechanism, it moves beyond simple data presentation. It offers a critical framework for evaluating the reliability of hybridization signals, distinguishing between localized noise and clade-wide consistent introgression events, and ultimately enabling a more nuanced and accurate interpretation of complex evolutionary histories.