README for Larson et al. (2021) Dryad submission

Article citation: Larson, D.A., Vargas, O.M., Vicentini, A. and Dick, C.W. (2021), Admixture may be extensive among hyperdominant Amazon rainforest tree species. New Phytologist. https://doi.org/10.1111/nph.17675

Dryad citation: Larson, Drew; Vargas, Oscar; Vicentini, Alberto; Dick, Christopher W. (2021), Admixture may be extensive among hyperdominant Amazon rainforest tree species, Dryad, Dataset, https://doi.org/10.5061/dryad.fj6q573t4

This README file provides information about the contents of the many files included in this repository. I have organized the files into directories based on the major kinds of analyses conducted in the study. However, these sections are not necessarily self-contained. For example, results in Tree_analyses/ were used to inform data subsetting in STRUCTURE/ and RT_tests/ uses alignments generated in the analyses associated with Tree_analyses/. See the publication and supporting information on more details about the various methods and workflows.

All code and scripts have been uploaded through a Zenodo submission via Dryad with the same directory structure described here. Therefore, any code or scripts are not contained within the Dryad download itself and should be accessed via Zenodo.

Below are links to each corresponding sections of this document.

Each item being explained is bolded. To limit the number of long pathnames, I use ">>>" to indicate the files and/or directories that follow are within this parent directory. When useful to understand what the files are, I have tried to include the exact code used to generate the files, though paths for some scripts and commands might no longer work. See the publication for the citation information for the programs and dependencies these scripts use. Unless noted, Python 2.7 was used to run all ".py" scripts.

If you use any of the scripts or data associated with this repository, please cite https://doi.org/10.1111/nph.17675. Please also cite any relevant dependencies used.

Last updated: September 9, 2021 Drew Larson

>>> Trimming_and_assembly/

Illumina_contaminants.fa

This file contains the sequences treated as contaminants (Illumina sequencing chemistry) during read trimming.

run_hybpiper_all_samples.sh

This bash script was used to run Hybpiper. Introns were subsequently recovered via command line, not as a script.

summary_sequence_lengths.txt

This file contains a summany of the sequence lengths generated with Hybpipers get_seq_lengths.py.

Lecythid_complete_targets

This is the target file used to assemble with Hybpiper. Targets are based on complete cDNA sequences identified by Vargas et al. (2019).

run_seqyclean.py
This script was used to trim forward and reverse reads prior to assembly with Hybpiper.

summary_statistics.txt

This file contains summary statistics on gene recovery created by Hybpiper.

>>> Tree_analyses

This folder contains the many files produced during the many tree-based analyses conducted including the Preliminary phylogeny and the Parvifolia phylogenies. Files include scripts, unaligned fasta files, alignments, lists, and phylogenies at many points along the workflow.

0a_exon_240/

This folder contains an unaligned fasta (directly from Hybpiper, no filtering) of exon sequences for each of the 343 loci for which exon sequences were recovered.

0b_intron_240/

This folder contains an unaligned fasta (directly from Hybpiper, no filtering) of intron sequences for each of the 343 loci for which intron sequences were recovered.

0c_aa_240/

This folder contains an unaligned fasta (directly from Hybpiper, no filtering) of amino acid sequences for each of the 343 loci for which amino acid sequences were recovered.

1a_exon_alignments/

This folder contains fasta files for exon sequences from 0a_exon_240 aligned with mafft --maxiterate 1000 --thread 15. These alignments are not filtered and were used only for visual inspection.

1c_aa_alignments/

This folder contains fasta files of amino acid sequences from 0c_aa_240 aligned with mafft --localpair --maxiterate 1000 --thread 10. These alignments were used to make amino acid trees for visual examination.

2a_exon_trees/

These are phylogenetic trees produced with RAxML for each of the exon alignments in 1a_exon_alignments/. They in preliminary analyses to determine whether to use nucleotide or amino acid trees to develop trimming cutoffs. Ultimately, amino acid trees were used.

They were produced with the following command:

for i in *FNA.mafft; do raxml -T 32 -m GTRCAT -p 12345 -s $i -n $i; done

2c_aa_trees/

These files are amino acid phylogenies for each of the 353 loci (1c_aa_alignments/) before any paralog filtering. Each was examined in order to identify which showed evidence of orthology issues.

These were produced with the command:
for i in *.localpair; do raxml -T 8 -m PROTCATWAG -p 12345 -s $i -n $i; done

>>> 3c_determine_cutoffs_aa/

This folder contains several files and scripts used to develop parameters for the first round of paralog trimming.

tree_validator.py

This script took unrooted trees output from RAxML and determined whether there as a tree bipartition that contained all outgroup samples. If there is no bipartiton with all the outgroups, that could be evidence of paralog or assembly issues. Trees in which all outgroup samples form a bipartition were considered valid and were saved in valid_aa_tree_files.txt.

This script was run as follows:
for i in 2c_aa_trees/RAxML_bestTree.AT*; do tree_validator.py $i; done > valid_aa_tree_files.txt

valid_aa_tree_files.txt

This file contains a list of paths to the 189 amino acid trees with all outgroup samples (species not in the Neotropical clade) forming a single bipartition, that is, those trees which passed the tree_validator.py script.

visual_inspection_trees.tre

This file contains 189 newick trees and was produced with the command: for i in $(cat valid_aa_tree_files.txt); do cat $i >> visual_inspection_trees.tre; done

results_visual_inspection.csv

This file is a csv where the first column is the path to a newick of an amino acid tree that was visually inspected and the second column contains a 0 for trees that contained suspicious internal branch lengths and a 1 for trees that did not. Each amino acid phylogeny in visual_inspection_trees.tre was visually inspected. The paths to amino acid trees that passed inspection are listed in results_visual_inspection_files.txt.

results_visual_inspection_files.txt

This is a file with 33 lines where aach line is a path to a newick file that passed manual visual inspection and has no apparent internal branch length issues that looked suspicious.

report_branchlengths.py

This script takes a newick tree an input and outputs a list of either the internal to terminal branch lengths to the screen. This was run on all amino acid trees to generate a file with all internal and all terminal branch lengths for all input trees in order to examine their distributions.

For example: for i in $(cat results_visual_inspection_files.txt); do python report_branchlengths.py internal $i; done

all_internal_branch_lengths_for_validated_trees.txt

This file contains a list of all internal branch lengths present in the amino acid trees that passed visual inspection. This file was produced with the following commands:

```
for i in $(cat results_visual_inspection_files.txt); do python report_branchlengths.py internal $i; done

cat 2c_aa_trees/*.internals.txt > all_internal_branch_lengths_for_validated_trees.txt
```

all_terminal_branch_lengths_for_validated_trees.txt

This file contains a list of all terminal branch lengths present in the amino acid trees that passed visual inspection. This file was produced with the following commands:

```
for i in $(cat results_visual_inspection_files.txt); do python report_branchlengths.py terminal $i; done
cat 2c_aa_trees/*.terminals.txt > all_terminal_branch_lengths_for_validated_trees.txt
```

4c_renamed_aa_trees/

This folder contains the same trees as in 2c_aa_trees/ but renamed with the following series of commands:

```
cp 2c_aa_trees/* 4c_renamed_aa_trees/
rename 's/RAxML_bestTree\.//g' RAxML_bestTree.AT*
rename 's/localpair/tre/g' AT*
```

5c_cut_internal_aa_trees/

This folder contains the 661 amino acid phylogenies after cutting internal branches longer than 0.25. The input trees were those in 4c_renamed_aa_trees/. The script used to trim was cut_long_internal_branches.py from Yang and Smith (2014).

The command was: python cut_long_internal_branches.py 4c_renamed_aa_trees/ .FAA.tre 0.25 100 5c_cut_internal_aa_trees/

6c_cut_terminal_aa_trees/

This folder contains the 661 amino acid phylogenies after cutting terminal branches longer than 0.15. The input trees were those in 5c_cut_internal_aa_trees/. The script used to trim was trim_tips.py from Yang and Smith (2014).

The command was:

```
python trim_tips.py 5c_cut_internal_aa_trees/ .subtree 20 0.15
```

7a_exon_fastas_filtered_based_on_aa/

This folder contains 661 fasta files for exon sequences produced by the script fastas_from_trees.sh. Each fasta is filtered to contain only the sequences that correspond to the tips remaining after trimming (those in the trees in 6c_cut_terminal_aa_trees/)

7b_intron_fastas_filtered_based_on_aa/

This folder contains 661 fasta files for intron sequences produced by the script fastas_from_trees.sh. Each fasta is filtered to contain only the sequences that correspond to the tips remaining after trimming (those in the trees in 6c_cut_terminal_aa_trees/)

7c_filtered_fastas_aa/

This folder contains 661 fasta files for amino acid sequences produced by the script fastas_from_trees.sh. Each fasta is filtered to contain only the sequences that correspond to the tips remaining after trimming (those in the trees in 6c_cut_terminal_aa_trees/)

8a_exon_filtered_localpair_alignments/

This folder contains the fastas for the sequences in 7a_exon_fastas_filtered_based_on_aa after alignment with mafft --localpair --maxiterate 1000. The *.filtered.localpair files are the output from mafft, while the *.filtered.localpair.pxclsq.30.aln files are the same alignments after removing columns with less than 30% occupancy with the phyx command pxlssq.

8b_intron_filtered_localpair_alignments/

This folder contains the fastas for the sequences in 7b_intron_fastas_filtered_based_on_aa after alignment with mafft --localpair --maxiterate 1000. The *.filtered.localpair files are the output from mafft, while the *.filtered.localpair.pxclsq.30.aln files are the same alignments after removing columns with less than 30% occupancy with the phyx command pxlssq.

8c_filtered_localpair_alignments_aa/

This folder contains the fastas for the sequences in 7c_filtered_fastas_aa after alignment with mafft --localpair --maxiterate 1000. The *.filtered.localpair files are the output from mafft, while the *.filtered.localpair.pxclsq.30.aln files are the same alignments after removing columns with less than 30% occupancy with the phyx command pxlssq. One fasta failed to align with the --localpair option and was instead aligned with the FFT-NS-i algorithm (i.e. AT1G79190.introns.fasta.AT1G79190_1.subtree.tt.filtered)

>>> **9_Preliminary_phylogeny_supermatix_exon_intron/**

This folder contains the input and output files used to produce the Preliminary phylogeny. The raxml output files are the result of the following commands:

```
raxml -T 16 -p 12345 -m GTRCAT -q Intron_Exon_supermatrix_005.model -s Intron_Exon_supermatrix_005.fa -n Intron_Exon_supermatrix_005_raxml

raxml -T 30 -p 12345 -m GTRCAT -x 12345 -# 200 -q Intron_Exon_supermatrix_005.model -s Intron_Exon_supermatrix_005.fa -n Intron_Exon_supermatrix_007_raxml_200rapidboots

raxml -m GTRCAT -f b -n Intron_Exon_supermatrix_008_raxml_200rapidboots -z RAxML_bootstrap.Intron_Exon_supermatrix_007_raxml_200rapidboots -t RAxML_bestTree.Intron_Exon_supermatrix_005_raxml
```

input_alignmnets/

This folder contains the intron and exon sequences from 8a_exon_filtered_localpair_alignments/ and 8b_intron_filtered_localpair_alignments/, which were used to make the supermatrixes.

Intron_Exon_supermatrix_005.fa
Intron_Exon_supermatrix_005.model

These files are the supermatrix and partition file used to generate the Preliminary phylogeny. They were generated using phyx and the command: pxcat -s *.pxclsq.30.aln -p Intron_Exon_supermatrix_005.model -o Intron_Exon_supermatrix_005.fa

>>> **10_Parvifolia_tree_orthogroups/**

This folder contains several directories created during the second round of trimming to ultimately produce the Parvifolia phylogeny. The samples included in the Parvifolia tree (including outgroups) are listed in nonHybrid_Parvifolia_taxa.txt.

10_Parvifolia_tree_orthogroups/1a_exon_alignments/

This folder contains the alignments from Tree_analyses/1a_exon_alignments/ but with any taxon that was not a member of the Parvifolia phylogeny in the Preliminary analysis (or one of the five included outgroup samples) removed. Inferred hybrids were also removed. Filenames were also shortened. Each of the resulting fastas is sorted into two folders based on whether they had 27 or fewer sequences (fail_occupancy) or more than that (pass_occupancy).

The following command was used for this step:
```
for i in *.localpair; do pxrms -s $i -f ../14_Parvifolia_tree/nonHybrid_Parvifolia_taxa.txt -c -o ../14_Parvifolia_tree/exon_fastas/$i.Parvifolia_tree.fasta; done
```

Passing alignmets were renamed with:
```
rename  's/subtree.tt.filtered.localpair/exon/g' *
```

10_Parvifolia_tree_orthogroups/1b_intron_alignments/

This folder contains the alignments from 8a_exon_filtered_localpair_alignments but with any taxon that was not a member of the Parvifolia phylogeny in the Preliminary analysis (or one of the five included outgroup samples) removed. Inferred hybrids were also removed. Filenames were also shortened. Each of the resulting fastas is sorted into two folders based on whether they had 27 or fewer sequences (fail_occupancy) or more than that (pass_occupancy).

Passing alignments were renamed with:
rename 's/subtree.tt.filtered.localpair/intron/g' *

10_Parvifolia_tree_orthogroups/2a_exon_cleaned_alignments/

These are the alignments from 10_Parvifolia_tree_orthogroups/1a_exon_alignments/pass_occupancy/ but after removing running pxclsq -p 0.3.

10_Parvifolia_tree_orthogroups/2b_intron_cleaned_alignments/

These are the alignments from 10_Parvifolia_tree_orthogroups/1b_intron_alignments/pass_occupancy/ but after running pxclsq -p 0.3.

10_Parvifolia_tree_orthogroups/3_parvifolia_orthogroup_trees/

This folder contains the results of running iqtree on all the alignments in 10_Parvifolia_tree_orthogroups/2a_exon_cleaned_alignments/ and 10_Parvifolia_tree_orthogroups/2b_intron_cleaned_alignments/ like:

iqtree -m GTR+G -s

10_Parvifolia_tree_orthogroups/4_trimmed_parvifolia_orthogroup_trees/

This folder contains the results of running trim_trees_based_on_branch_distributions.py on all trees in 10_Parvifolia_tree_orthogroups/3_parvifolia_orthogroup_trees/. Due to a file naming issue, the intron and exon trees were split before running trim_trees_based_on_branch_distributions.py, then processed with trim_trees_based_on_branch_distributions.py and renamed to specify whether each was an intron or exon tree.

10_Parvifolia_tree_orthogroups/5_trimmed_parvifolia_orthogroup_alignments/

This folder contains an alignment based on the final tips each of the trimmed trees in 10_Parvifolia_tree_orthogroups/4_trimmed_parvifolia_orthogroup_trees/. Separately, for introns and exon the following commands were run:

bash make_exon_Parvifolia_alignments_from_trees.sh
bash make_intron_Parvifolia_alignments_from_trees.sh

note: the paths referenced by these two scripts have changed slightly, since the contents of trimmed_orthogroup_exons/ and trimmed_orthogroup_introns/ are both now in 5_trimmed_parvifolia_orthogroup_alignments.

Then the following commands were run to remove any sequences that would have more than 75% missing data after pxclsq -p 0.3.

```
for i in *; do pxclsq -p 0.3 -s $i -o $i.30.pxclsq; done

for i in *.pxclsq; do python ../count_missing_data_in_align.py $i; done

for i in *.localpair; do pxrms -s $i -f $i.30.pxclsq.to_remove -o $i.cleaned; done

for i in *.cleaned; do pxclsq -p 0.3 -s $i -o $i.30.pxclsq; done
```

10_Parvifolia_tree_orthogroups/6_supermatrix_inputs/

The files in this folder are the alignments output by the final command run in 5_trimmed_parvifolia_orthogroup_alignments/, which are named like *.subtree.tt.parvifolia.localpair.cleaned.30.pxclsq. Any alignment that had fewer than 27 taxa (25% of the total number of samples included in the Parvifolia tree sampling) were deleted at this point.

Files were renamed with the following command

```
rename 's/\.subtree\.tt\.parvifolia\.localpair\.cleaned\.30\.pxclsq/\.localpair/g' *
```

Which for example changed:
AT3G18390_2_exon_1.subtree.tt.parvifolia.localpair.cleaned.30.pxclsq
To
AT3G18390_2_exon_1.localpair

11_Parvifolia_tree/

This folder contains the inputs and outputs of the Parvifolia tree tree-searches.

The commands to produce the supermatrixes and partition files from 10_Parvifolia_tree_orthogroups/6_supermatrix_inputs/ are:

```
pxcat -s *.localpair -o Parvifolia_intronexon_supermatrix.fa -p
Parvifolia_intronexon_supermatrix.model

pxcat -s *exon *.localpair -p ../Parvifolia_tree_exon_only.model -o
../Parvifolia_tree_exon_only_supermatrix.fa
```

11_Parvifolia_tree/Tree_searches/

This folder contains all of the output files from the various tree-searches outlined in the methods. The exact commands used can be found in the log files.

11_Parvifolia_tree/lk_AC_parvifolia/

This folder constains all the output files generated by re-computing likelihoods and information criterion scores with iqtree. The exact commands used can be found in the log files and the script run_lk_AC.sh.

>>> 12_Parvifolia_tree_reduced_tips/

This folder contains some files used in generating the reduced tips version of the Parvifolia trees.

Iqtree_Parvifolia_exon_GTRG.treefile

This file is the Exon-only Parvifolia tree determined to be the best based on AIC score.

Iqtree_Parvifolia_intronexon_GTRG.treefile

This file is the Parvifolia tree determined to be the best based on AIC score.

Parvifolia_tree_representative_tips.txt

This is a file that contains the sample names to keep in the representative tree.

Change_labels_to_species_names_parvifolia_tree.py

This script takes a tree as sys.argv[1], finds Code_to_species_names.csv and changes the sample label names to species names based on the csv.

Code_to_species_names.csv

This file is a csv with two columns. The first is the sample code and the second is the species name to be included in the reduced representation trees.

exon_reduced_tips.tre

The exon-only Parvifolia tree reduced to representative tips with:
pxrmt -f Parvifolia_tree_representative_tips.txt -t Iqtree_Parvifolia_exon_GTRG.treefile -o exon_reduced_tips.tre -c

exon_reduced_tips.tre.rr

The file exon_reduced_tips.tre rooted on the outgroup EsinLA01.

exon_reduced_tips.tre.rr.names

The file exon_reduced_tips.tre.rr with species name instead of labels through:
python Change_labels_to_species_names_parvifolia_tree.py exon_reduced_tips.tre.rr

intronexon_reduced_tips.tre

The Parvifolia tree reduced to representative tips with:
pxrmt -f Parvifolia_tree_representative_tips.txt -t Iqtree_Parvifolia_intronexon_GTRG.treefile -o intronexon_reduced_tips.tre -c

intronexon_reduced_tips.tre.rr

The file intronexon_reduced_tips.tre rooted on outgroup EsinLA01

intronexon_reduced_tips.tre.rr.names

The file **intronexon_reduced_tips.tre.rr** with species name instead of labels through:
python Change_labels_to_species_names_parvifolia_tree.py **intronexon_reduced_tips.tre.rr**

intronexon_reduced_tips.tre.rr.names.pxbpmapped.tre

This file is the Parvifolia tree with reduced representation where node labels indicate conflict with the reduced representation Exon-only Parvifolia phylogeny.

Produced with the command:
pxbp -m intronexon_reduced_tips.tre.rr.names -t exon_reduced_tips.tre.rr.names

cophylo_representative_tips_plot.R

This is an R script that plots the reduced representation Parvifolia phylogenies using the phytools library.

>>> **Tree_analyses/scripts/**

fasta_from_tree.py

This script takes a unfiltered fasta and a filtered (i.e. trimmed) tree as input and outputs a filtered fasta based on which tips are present in the tree.

run like: python fasta_from_tree.py unfiltered_fasta filtered_tree

generate_fasta_from_tree_script.py
This script generates a bash script that runs fasta_from_tree.py for all samples. The output of this script is fastas_from_trees.sh.

fastas_from_trees.sh

This is a bash script that runs fasta_from_tree.py for all samples. A separate version was used for exons, introns, and amino acid sequences.

parvifolia_alignment_from_tree.py

This script
run like python parvifolia_alignment_from_tree.py unfiltered_fasta filtered_tree

generate_parvifolia_alignment_from_tree.py

A python script to generate a bash script to run parvifolia_alignment_from_tree.py for all orthogroups. The output of this command is Parvifolia_alignments_from_trees.sh which can consist of either exon or intron data depending on which unfiltered_fasta_dir is specified in the code.

make_exon_Parvifolia_alignments_from_trees.sh

This script is the result of running generate_parvifolia_alignment_from_tree.py for exons based on the trimmed trees in **4_trimmed_parvifolia_orthogroup_trees/**. The output filepath is different than the current name of the folder containing the output from this script, which is now 5_trimmed_parvifolia_orthogroup_alignments.

make_intron_Parvifolia_alignments_from_trees.sh

This script is the result of running generate_parvifolia_alignment_from_tree.py for introns based on the trimmed trees in **4_trimmed_parvifolia_orthogroup_trees/**. The output filepath is different than the current name of the folder containing the output from this script, which is now 5_trimmed_parvifolia_orthogroup_alignments.

count_missing_data_in_align.py

This script takes an alignment as input, and calculates the amount of missing data, or inferred gaps for each sample. Any sample with missing data greater than the threshold is recorded in an output file which can be used to remove those taxa from the alignment using phyx.

For example:

```
for i in *.pxclsq; do python ../count_missing_data_in_align.py $i; done
```

which produces files like *.pxclsq.to_remove, which were used to run phyx like:

```
for i in *.localpair; do pxrms -s $i -f $i.30.pxclsq.to_remove -o $i.cleaned; done
```

>>> STRUCTURE/

Each of the folders in this directory contains the files associated with a STRUCTURE dataset. The samples included in each dataset can be found in the first table of the supporting information of the publication. See the section on convert_plinkRecode_to_structure.py in this document and supporting methods in the publication for more information in each of these files.

The parameters_input_and_output/ folder contains the actual input and output files for STRUCTURE. The output files were formatted with f2R.py prior to plotting in R – see the notes on that script in this document for more information.

The file parameters_input_and_output/mainparams contains the main parameters used by STRUCTURE. A single mainparams file per dataset is included in the repository as only the number of populations (K) was altered between runs for the same dataset. A list of all Ks run for each dataset is reported in the supporting information.

>>> STRUCTURE/scripts

calc_missing_data_structure.py

This script takes a STRUCTURE output file (e.g. OAP/parameters_input_and_output/OAP_1pop_f) as sys.argv[1] and reports the following, which are documented in the supporting information for each dataset:

1)The average missing data per locus
2)The number of loci
3) The number of loci with no missing data

convert_plinkRecode_to_structure.py

This script takes the output of "plink --recode structure", formats the file, and adds population (i.e. species) data specified.

The input to this script was generated as follows:

1) plink --vcf ../Genotypes_110_parvifolia_clade_EscoL834_ref_SNP_only.vcf --keep X.plink.txt --allow-extra-chr --make-bed
2) plink2 --allow-extra-chr --bfile plink --set-all-var-ids @_#_\$r_\$a --out prefilter --new-id-max-allele-len 286 --max-alleles 50 --make-bed
3) plink2 -bfile prefilter --indep-pairwise 50kb 1 0.0001 --allow-extra-chr -out LD_STEP
4) plink2 --allow-extra-chr --bfile prefilter --out final --new-id-max-allele-len 286 --max-alleles 50 --make-bed --extract LD_STEP.prune.in
5) plink --recode structure --allow-extra-chr -bfile final

The script was run like:
python convert_plinkRecode_to_structure.py plink.recode.strct_in X.pops.txt

X.pops.txt is a comma-delimited file were the first column is the sample name and the second column is the population (species) to which that sample is thought to belong. For example:

```
EswaL695,2
EswaL832,2
EscoL796,1
EscoL824,1
EscoL770,1
EscoL780,1
```

X.plink.txt is a file that lists the samples to include in the subset in the following format: sample_label(tab)sample_label. For example:

```
EstrL772	EstrL772
EsbrL733	EsbrL733
EstrL802	EstrL802
EswaL839	EswaL839
EsteL690	EsteL690
EstrL698	EstrL698
```

X.plink.txt and X.pops.txt should include the same samples and there should be one sample per line in each file.

f2R.py

This script takes an output file from STRUCTURE and generates a new file formatted to be read by the R script R_scripts_and_plotting/Code_Figs_2_3_S3_S4_S5_astrisks.R. The script name means “format to R”.

sys.argv[1] should the structure output file
sys.argv[2] should be the number of populations (K=) with which you ran structure
sys.argv[3] is the output csv file that will be created with columns:
Population,Label,Cluster1,Cluster2,Cluster3,etc

find_inds_and_loci.py

This script takes a STRUCTURE input file (e.g. /OAP/plink.recode.strct_in.structure_input_formatted.str) as sys.argv[1] and outputs the number of loci and number of individuals in the file, which is useful when making the STRUCTURE config file.

>>> SNP_calling/

110_samples_included.txt

This file contains the codes for the 110 samples for which SNPs were called.

Genotypes_110_parvifolia_clade_EscoL834_ref_SNP_only.vcf

This file is the vcf file resulting from the SNP-calling pipeline, which was used to produce the datasets for STRUCTURE analyses.

>>> SNP_calling/reference

This folder contains the reference genome fasta file and the various index files used in a number of analyses.

>>> SNP_calling/scripts/

Count_total_polymorphic_sites.py

This script simply counts the number of polymorphic sites in a vcf file containing the 109 members of the Parvifolia clade.

Count_total_reference_length.py

This script counts the number of "Ns" and nucleotides in the reference genome used in this study. It also checks the number of contigs in the reference.

GATK_sam_to_halplotypeCaller_parvifolia.py

This script conducts several steps of the pipeline starting with a sam file for a sample, processing them, and running Haplotype caller to produce a g.vcf for the sample.

combine_reads_capt_WGS.py

This script was used to combine target capture sequencing reads with those from whole genome sequencing of unenriched Illumina libraries.

generate_CombineGVCF_command.py

This script was used to format the command used to run CombineGVCFs.

generate_sam_files_from_raw_reads_parvifolia.py

This script runs bwa mem for all samples to produce a sam file from raw forward and reverse reads.

make_reference_genome_from_exonerate_exons.py

This script was used to produce the reference genome for a given sample using the directory structure output by Hybpiper. The sample EscoL834 was used for downstream analyses.

parvifolia_combine_command.gakt

This is the bash command output from generate_CombineGVCF_command.py. It runs CombineGVCFs in GATK.

>>> RT_tests/

Format_RT_results/

This folder contains various scripts and files used to format the results of the RT tests (i.e. RT_results_summary.tsv) for publication.

Run_RT_tests.py

This python script takes an RT test input file and runs an RT test for each test specified. This script is run like:
python Run_RT_test.py input_file.

all_RT_input_alignments.tar.gz

This is a gzipped tar archive containing the alignments used for RT testing. There are multiple alignments for a single named locus (e.g. AT5G26850) if the sequences recovered for that locus were split during the tree-based ortholog trimming procedure. Intron and exon sequences for each orthogroup are in separate alignments. Sequences from locus AT1G79190 were not included, because the intron sequences for this locus failed to align with the L-INS-i algorithm in MAFFT.

The exon alignments are those from /Tree_analyses/8a_exon_filtered_localpair_alignments with names ending in .subtree.tt.filtered.localpair.pxclsq.30.aln but which have been renamed with the following commands:

```
rename 's/.*\.FNA\.//g' *.aln
rename 's/\.subtree\.tt\.filtered\.localpair\.pxclsq\.30\.aln/\.exon.aln/g' *.aln
```

The intron alignments are those from /Tree_analyses/8b_intron_filtered_localpair_alignments with names ending in *.subtree.tt.filtered.localpair.pxclsq.30.aln but which have been renamed with the following commands:

```
rename 's/.*\.introns\.fasta\.//g' *.aln
rename 's/\.subtree\.tt\.filtered\.localpair\.pxclsq\.30\.aln/\.intron.aln/g' *.aln
```

completed_tests.tar.gz

This is a gzipped tar achieve containing key output files generated by iqtree in RT tests in the study. It does not contain the many sub-alignments generated, which were deleted with clean_up_RT_files.py.

RT_results_summary.tsv

This file is the output of Summarize_list_of_RT_runs.py before it was augmented with additional taxonomic information for the samples, which appears in the publication.

Summarize_list_of_RT_runs.py

This script uses the input file used in Run_RT_tests.py and summarizes the results of all runs in the input file. Run like: python Summarize_list_of_RT_runs.py inputfile.

clean_up_RT_files.py

This script moves the sub-alignments produced by Run_RT_tests.py and transfers them to a new directory where they can be stored or deleted. Run like: python clean_up_RT_files.py input_file.txt dir_for_files.

input_45_tests.txt

This is the input file used to run the RT tests in this study. It is a tab-separated file where the first column is the user-specified name for the test, the second column is a comma-separated list of three ingroup individuals for the test, and the third column is the single outgroup individual.
For example:

```
1_OCWT	EswaL779,EscoL834,EstrL838	EsinLA01
```

>>> Phenology/

Formatted_flowering_collections.csv

This file contains a CSV of unique specimen collection dates for E. coriacea, E. wachenheimii, and E. parviflora, after removing duplicates collected on the same day from the same tree. These records are from collections at the C. V. Starr Virtual Herbarium of the New York Botanical Garden Herbarium accessed in January of 2021.

Phenology_boxplot_script.R

This is the R script used to make boxplots of flowering times.

>>> R_scripts_and_plotting/

Code_Figs_2_3_S3_S4_S5_astrisks.R

This R script contains the code to produce the STRUCTURE and PCA plots used to create the figures.

Coriacea_PCA/

This folder contains .bed and associated files used to produce the PCA of 12 E. coriacea samples.

Formatted_structure_results/

This folder contains the results of all STRUCTURE analyses, after processing the output file with the script STRUCTURE/scripts/f2R.py. These are the files that Code_Figs_2_3_S3_S4_S5_astrisks.R needs to produce the plots.

>>>Map

Parvifolia_map.qgs.qgz

This is the QGIS file used to produce the map of collection records for species of the Parvifolia clade. The map was produced with QGIS v3.16.3.

Lecythidaceae_updated/

The actual shapefile and associated files are not included in the Dryad submission.

This GIS layer contains Lecythidaceae collection records curated by Mori et al. (2017).

Mori SA, Kiernan EA, Smith NP, Kelly LM, Huang Y-Y, Prance GT, Thiers, BM. **2017**. Observations on the phytogeography of the Lecythidaceae clade (Brazil nut family). *Phytoneuron* **30**: 1–85.

DEM_30s_BIL/

This GIS layer contains the Digital Elevation Model data layer used in the map.

The actual shapefile and associated files are not included in the Dryad submission.

Lehner, B., Grill G. (2013): Global river hydrography and network routing: baseline data and new approaches to study the world's large river systems. Hydrological Processes, 27(15): 2171–2186.

License agreement: https://www.hydrosheds.org/pages/license
Accessed from: https://www.hydrosheds.org/downloads

TM_WORLD_BORDERS-0/

This GIS layer contains the national boarders used in the map.

The actual shapefile and associated files are not included in the Dryad submission.

World Borders Dataset provided by Bjorn Sandvik, thematicmapping.org
Used under the terms of the Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) license: https://creativecommons.org/licenses/by-sa/3.0/

Accessed from: http://thematicmapping.org/downloads/world_borders.php

"The original shapefile (world_borders.zip, 3.2 MB) was downloaded from the Mapping Hacks website: http://www.mappinghacks.com/data/"

majorrivers_0_0/

This GIS layer contains the river data used in the map.

The actual shapefile and associated files are not included in the Dryad submission.

This dataset is licensed under CC-BY 4.0
The World Bank
Accessed from: https://datacatalog.worldbank.org/dataset/major-rivers-world

"Limitations and Exceptions:
Keith Patrick Garrett/GOST is not necessarily the originator of this data, if you have questions about this data please contact gost@worldbank.org"

manaus/

This GIS layer contains one data point, the estimated location of Manaus, Brazil. The city's location was estimated, and the layer was produced by Drew Larson.