

# Brief intro to the **phruta** R package

Cristian Roman-Palacios

First created on 2021-07-30. Updated on 2021-08-01

## 1 Table of Contents

1. What is **phruta**
2. Functions in **phruta**
3. From clade names to time-calibrated phylogenies
4. Installing RAxML
5. Installing PATHd-8 and treePL

### 1.1 What is **phruta**

The **phruta** R package is designed to simplify the basic phylogenetic pipeline. Specifically, all code is run within the same program and data from intermediate steps are saved in independent folders. Furthermore, all code is run within the same environment which increases the reproducibility of your analysis. **phruta** retrieves gene sequences, combines newly downloaded and local gene sequences, and performs sequence alignments.

**phruta** is also able to perform basic phylogenetic inference under **RAxML** on the resulting sequence alignments. The current release allows users to conduct tree dating based on secondary calibrations. **phruta** is essentially a wrapper for alternative R packages and software.

### 1.2 Functions in **phruta**

The current release of **phruta** includes a set of six major functions. All six functions form a pipeline within **phruta** to output a time-calibrated phylogeny. However, users interested in using their own files at any stage can run each function independently.

Note that all the functions for which their primary output are sequences (aligned or unaligned) are listed under **sq.\***. All the files that output phylogenies (time-calibrated or not) are listed under **tree.\***.

- First, **sq.retrieve** retrieves gene sequences for taxa of interest. This function saves all the resulting **fasta** files in a new directory, **0.Sequences**.
- Second, **sq.add** allows users to include local sequences to those retrieved from genbank in the previous step. This function saves all the resulting **fasta** files in two directories, combined sequences in **0.Sequences** and local sequences in **0.AdditionalSequences** (originally downloaded sequences are moved to **0.0.OriginalDownloaded** at this step). Note that **sq.add** is optional.
- Third, the **sq.curate** function filters out unreliable sequences based on information listed in genbank (e.g. **PREDICTED**) and on taxonomic information provided by the user. Specifically, this function retrieves taxonomic information from the Global Biodiversity Information Facility (GBIF) database's taxonomic backbone. If a given species belongs to a non-target group, this species is dropped from the analyses. This function automatically corrects taxonomy and renames sequences.

- Fourth, `sq.aln` performs multiple sequence alignment on `fasta` files. Currently, `phruta` uses the DECIPHER R package, here. This package allows for adjusting sequence orientation and masking (removing ambiguous sites).

The final two functions in `phruta` focus on tree inference and dating. These two functions depend on external software that needs to be installed (**and tested**) before running. Please make sure both `RAxML` and `PATHd-8` or `treePL` are installed and can be called within R using the `system()` function. Note that you can choose between `PATHd-8` and `treePL`. More details on how to install `RAxML` are provided below. Similarly, we provide details on how to install `PATHd-8` and `treePL` below.

-Fifth, the `tree.raxml` function allows users to perform tree inference under `RAxML` for sequences in a given folder. This is a wrapper to `ips::raxml` and each of the arguments can be customized. The current release of `phruta` can manage unpartitioned analyses without starting trees.

-Sixth, `tree.dating` enables users to perform time-calibrations of a given phylogeny using `geiger::congruify.phylo`. `phruta` includes a basic set of comprehensively sampled, time-calibrated phylogenies that are used to extract secondary calibrations for the target phylogeny. Note that sampling in those phylogenies can be examined using `data(SW.phruta)`. Please make sure you have at least **two** groups in common with each of the phylogenies. Similarly, users can choose to run either `PATHd-8` or `treePL`.

### 1.3 Tutorial: Clade names to time-calibrated phylogenies

Let's create a phylogeny for a few mammal clades. Assume that we need to build a tree for the following three genera: *Felis*, *Vulpes*, and *Phoca*. Note that all three genera are within Carnivora. Both *Felis* and *Vulpes* are classified in different superfamilies within Fissipedia and *Phoca* is part of another suborder, Pinnipedia. We're going to root our tree with another mammal species, a Chinese Pangolin (*Manis pentadactyla*). Users can select additional target species, however, for simplicity, we will run the analyses using three genera in the ingroup and a single outgroup species.

So far, we have decided the taxonomic make of our analyses. However, we also need to determine the characteristics of the molecular dataset that we would like to assemble. Fortunately, mammals are extensively studied and a comprehensive list of potential gene regions to be analyzed is already available. I will use the same gene regions as in the excellent Upham et al (2009).

Before we move on, please make sure that you have set a working directory for this project as all the files will be saved to the current working directory.

Now, since we're going to retrieve sequences from genbank, we will use the `sq.retrieve` function in `phruta`. This function will create `fasta` files (saved to your working directory) for the genes where at least one sequence was found in genbank for the target taxa.

```
sq.retrieve(
  clades = c('Felis', 'Vulpes', 'Phoca'),
  species = 'Manis_pentadactyla',
  genes = c("A2AB", "ADORA3", "ADRB2", "APOB",
            "APP", "ATP7", "BCHE", "BDNF",
            "BMI1", "BRCA1", "BRCA2", "CNR1",
            "COI", "CREM", "CYTB", "DMP1",
            "EDG1", "ENAM", "FBN1", "GHR",
            "IRBP", "ND1", "ND2", "PLCB4",
            "PNOC", "RAG1a", "RAG1b", "RAG2",
            "TTN", "TYR1", "VWF")
)
```

This function will write all the resulting `fasta` files into the newly created folder `0.Sequences` located in our working directory.

Next, we're going to make sure that we include only sequences that are reliable and from species that we are interested in analyzing. We're going to use the `sq.curate` function for this. We will provide a list of taxonomic criteria to filter out incorrect sequences (`filterTaxonomicCriteria` argument), which in this case can be just the genera that we're interested in analyzing. Note that the outgroup's name is also included in the list. If the taxonomic information for a sequence retrieved from genbank does not match with any of these strings, this species will be dropped. You will have to specify whether sampling is for animals or plants (`kingdom` argument). Hopefully, I will be able to expand this to other kingdoms soon.

```
sq.curate(filterTaxonomicCriteria='Felis|Vulpes|Phoca|Manis',
          kingdom='animals',
          folder='0.Sequences')
```

Running this line will create the new folder `1.CuratedSequences` that contains (1) the curated sequences with original names, (2) curated sequences with species-level names (`renamed_*` prefix), (3) an accession table (`0.AccessionTable.csv`), and (4) a summary of taxonomic information for all the species sampled in the files (`1.Taxonomy.csv`). We'll use the `renamed_*` and `1.Taxonomy.csv` files in the next steps.

Next, we'll align the sequences that we just curated. For this, we just use `sq.aln` with default parameters.

```
sq.aln(folder='1.CuratedSequences')
```

The resulting multiple sequence alignments will be saved to the `2.Alignments` folder. In this folder, we will have two types of files: (1) raw alignments (same file names as in `1.CuratedSequences`) and (2) alignments with ambiguous sites removed (`Masked_*` prefix). We will use the latter set of files in the next steps.

Phylogenetic inference is conducted using the `tree.raxml` function. We need to indicate where the aligned sequences are located (`folder` argument), the patterns of the files in the same folder (`FilePatterns` argument; "Masked" in our case). We'll run a total of 100 bootstrap replicates and set the outgroup to "Manis\_pentadactyla".

```
tree.raxml(folder='2.Alignments',
           FilePatterns= 'Masked',
           raxml_exec='raxmlHPC',
           Bootstrap=100,
           outgroup ="Manis_pentadactyla")
```

The trees are saved in `3.Phylogeny`. Likely, the bipartitions tree, "RAxML\_bipartitions.phruta", is the most relevant. `3.Phylogeny` also includes additional RAxML-related input and output files.

Finally, let's perform tree dating in our phylogeny using secondary calibrations extracted from Scholl and Wiens (2016). This study curated potentially the most comprehensive and reliable set of trees to summarize the temporal dimension in evolution across the tree of life. In `phruta`, the trees from Scholl and Wiens (2016) were renamed to match taxonomic groups.

Tree dating is performed using the `tree.dating` function in `phruta`. We have to provide the name of the folder containing the `1.Taxonomy.csv` file created in `sq.curate`. We also have to indicate the name of the folder containing the `RAxML_bipartitions.phruta` file. We will scale our phylogeny using `treePL`.

```
tree.dating(taxonomyFolder="1.CuratedSequences",
            phylogenyFolder="3.Phylogeny",
            scale='treePL')
```

Running this line will result in a new folder `4.Timetree`, including the different time-calibrated phylogenies obtained (if any) and associated secondary calibrations used in the analyses. We found only a few overlapping calibration points (family-level constraints):



Open R and make sure that the following line doesn't throw an error.

```
system("raxmlHPC")
```

You may also want to check if RAxML is called using `raxmlHPC` or something else. This can be set when running `tree.raxml` using the `raxml_exec` argument.

Finally, note that RStudio sometimes has issues finding *stuff* in the path while using `system()`.

With issues, if you're running macOS, try running RStudio from the command line. Close RStudio, open the terminal, and type:

```
open /Applications/RStudio.app
```

## 1.5 Installing PATHd-8 and treePL

There are excellent guides for installing PATHd-8 and treePL. Here, I summarize two potentially relevant options.

First, you can use Brian O'Meara's approach for installing PATHd-8. I summarized the code here.

Second, you can use homebrew to install treePL, thanks to Jonathan Chang.

```
brew install brewsci/bio/treepL
```

## 1.6 Future developments

I acknowledge that the current release of the package is limited in different ways. Next releases will focus on improving the following aspects:

- Expand taxonomic cleaning for non-animal/plant groups
- Taxonomic tree constraints
- Starting trees in RAxML
- Partitioned analyses in RAxML
- PartitionFinder
- <https://github.com/ms609/Roguer/>

Please stay tuned for updates. Get in touch if you have questions. Open issues if you find anything relevant in my code, have suggestions, etc. Your input is very valuable!