Brief intro to the `phruta` R package

Cristian Roman-Palacios

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

1 Table of Contents

1.1 What is `phruta`

The `phruta` R package is designed to simplify the basic phylogenetic pipeline. Furthermore, the fact that all the intermediate steps are saved in independent folder and all the code is run within the same environment increases reproducibility. `phruta` retrieves gene sequences, combines newly downloaded and local gene sequences, and perform sequence alignment.

`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 different R packages and additional software.

1.2 Functions in `fruta`

The current release of `fruta` includes a set of six different major functions. If users are interested in using their own files at any stage, each of the functions can be used independently. However, there is a general sequence that can be used to generate a pipeline within `fruta`. 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.
- Second, `sq.add` allow users to include local sequences to the sequences that were retrieved from genbank in the previous step. Note that `sq.add` is optional. However, if this function is run, the user will have available the originally downloaded sequences, the local sequences, and the combined set of sequences in three independent folders.
- 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 gbif taxonomic backbone for each of the species in the dataset. If a given species belongs to a non-target group, this species is dropped from the analyses. This function automatically corrects taxonomy and rename sequences.
- Fourth, `sq.aln` performs multiple sequence alignment on `fasta` files. Currently, `phruta` uses the `DECIPHER` R package for this. This package allows for adjusting sequence orientation and masking.

The final two functions in `phruta` focuses on tree interence and dating. These two functions depend on external software that needs to be installed (and tested) before using them. Please make sure `RAxML`, `PATHd-8`, and `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 a sequences in a given folder. Note that users should provide a list of outgroup taxa and conduct a bootstrap analysis. This is a wraper to `ips::raxml` and all their arguments can be customized. The current release can only handle 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 comprehensenvisely 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 last *2* groups in common with any of the phylogenies. Finally, analyses can be run under either `PATHd-8` or `treePL`.

1.3 From clade names to time-calibred phylogenies

Let's create a phylogeny for a few mammal clades. We're going to 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 out tree with another mammal species, a Chinese Pangolin (*Manis pentadactyla*). Note that we can also select additional target species that can be or not of the outgroup. However, for simplicity, we will run the analyses using three genera in the outgroup and a single outgroup species.

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

Before we move on, please make sure that you you have set a working directory for this project. *Please do not forget to set the working directory to a particular folder*. All the files will be saved to the 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 a 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 and plants (`kingdom` argument). Hopefully, I will be able to expand this to other groups soon. Note that our downloaded sequences are located in the `0.Sequences` folder.

```
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. Note that all the target fasta files are in `1.CuratedSequences` and they all have the default prefix (`renamed_*`).

```
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. First, files that are just the raw alignments (same file names as in `1.CuratedSequences`). Second, sequence alignments that with ambiguous sites removed (`renamed_*` 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). Well run a total of `100` boostrap replicates and set the outgroup to `Manis_pentadactyla`.

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

The tree (probably the bipartitions is the most relevant one; `RAxML_bipartitions.phruta`) is saved under the `3.Phylogeny` folder. This folder also include additional `RAxML`-related input and output files. Finally, let's perform tree dating in our phylogeny using secondaty calibrations extracted from Scholl and Wiens (2016). Here's a link to the study. In short, this study potentially the most comprenhensive and reliable set of trees that summarize the temporal dimension in the evolution across the tree of life. In `phruta`, the trees in School 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 before. 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 obained (if any) and associated secondary calibrations used in the analyses. We found only a couple of overlapping calibration points (family-level constraints):

FIELD1	MRCA	MaxAge	MinAge	taxonA	taxonB
1	cc0dbab90e7361c3397d593bf51b72ab	79.4299999999994	79.4299999999994	Felis_manul	Manis_pentadactyla
2	fccec89202f09a5186e22a6d51ec9456	56.6899999999996	56.6899999999996	Felis_manul	Phoca_largha
8	5609448ff616023d7be2b0e15b7d6f5d	47.9699999999993	47.9699999999993	Phoca_largha	Vulpes_velox

Here's the resulting time-calibrated phylogeny. The whole process this took ~20 minutes to complete in my

computer (16 gb RAM, i5).

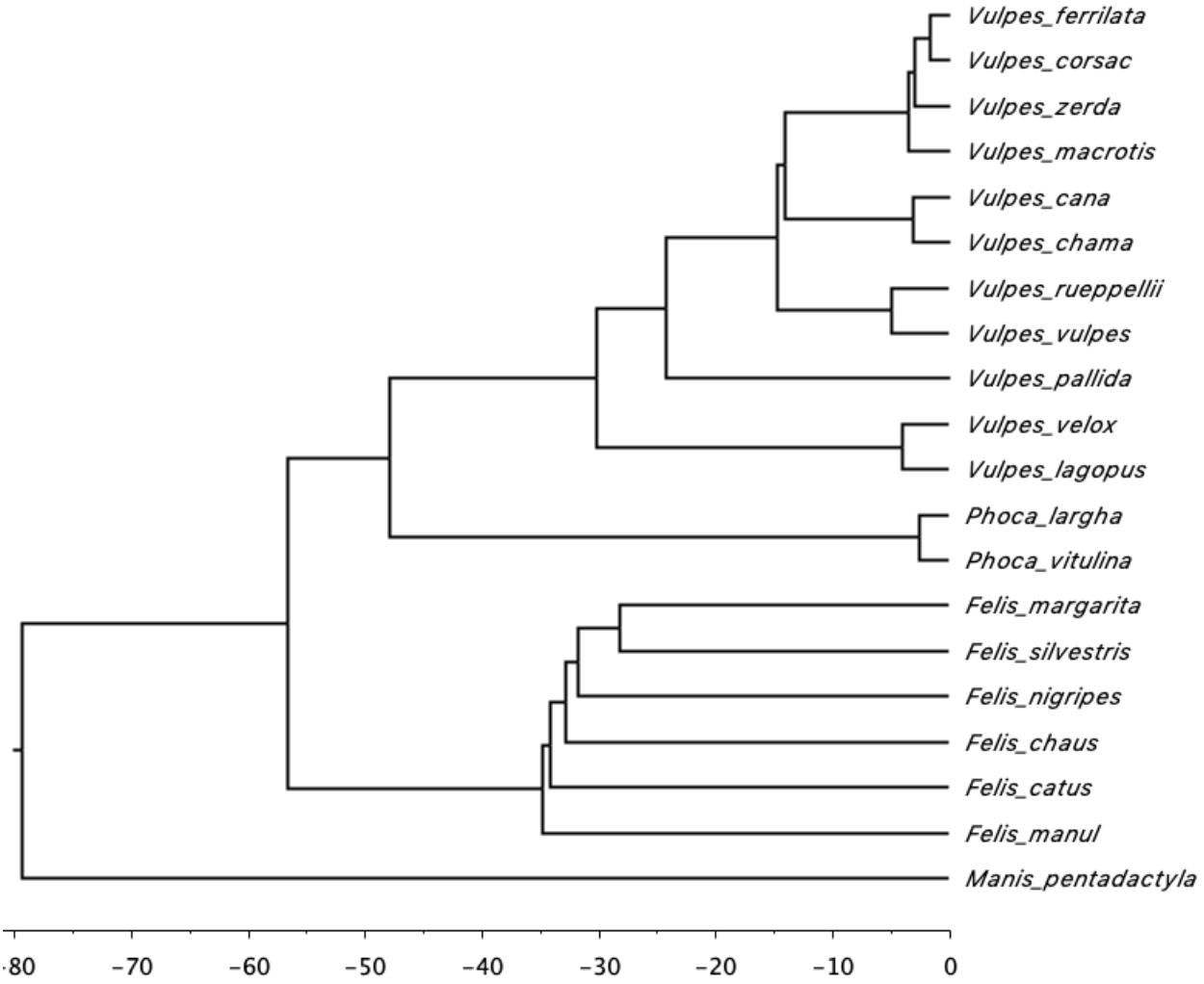

Figure 1: Time-calibrated phylogeny for the target taxa. Analyses performed within the phruta R package.

1.4 Installing RAxML

`RAxML` can be easily installed to the `PATH` using one of the two lines below under `conda`:

```
conda install -c bioconda raxml
conda install -c bioconda/label/cf201901 raxml
```

Open `r` and make sure that the following line does'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 runnig the in using the `tree.raxml` using the `raxml_exec` argument.

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

If you're using macOS + RStudio, close R studio, 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`. I just want to 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` (`brew install brewsci/bio/treepl`). This latter is thanks to Jonathan Chang

1.6 Future developments

I acknowledge that the current release of the package is limited in different ways. However, 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

Please keep tunned 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!