Package: phruta (via r-universe)

July 5, 2024

Type Package

Title Phylogenetic Reconstruction and Time-dating

Version 0.1.3

URL https://docs.ropensci.org/phruta/index.html,
https://github.com/ropensci/phruta,
https://ropensci.github.io/phruta/

BugReports https://github.com/ropensci/phruta/issues

Description 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.

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports rgbif, pbapply, rentrez, taxize, ips, ape, geiger, methods, DECIPHER, reutils, foreach, doParallel, doSNOW, Biostrings, msa, odseq, XML, Rogue

RoxygenNote 7.2.0

Suggests testthat (>= 3.0.0), covr, knitr, markdown, rmarkdown, spelling, utf8

VignetteBuilder knitr

License MIT + file LICENSE

Config/testthat/edition 3

Language en-US

Roxygen list(markdown = TRUE)

Repository https://ropensci.r-universe.dev

RemoteUrl https://github.com/ropensci/phruta

RemoteRef main

RemoteSha a664f5f2d9788e6417361b2e0c0ff2cff63f7d17

Contents

`acc.table.retrieve` *Retrieve accession numbers and titles for a given combination of species and genes in genbank*

Description

Retrieve accession numbers and titles for complex searches conducted in genbank. Note that this function depends on `acc.retrieve` as it actually uses `expand.grid` to find all the relevant organism/gene combinations in genbank.

Usage

```
acc.table.retrieve(
  clades = NULL,
  species = NULL,
  genes = NULL,
  speciesLevel = NULL,
  npar = 2,
  nSearchesBatch = 499
)
```

Arguments

`clades` A vector of clade names (character). Note that this can either be the name of a clade (e.g. Apis) or the code in NCBI (e.g. txid7459).

`species` A vector of species names (logical). Note that this can either be the name of a clade (e.g. Apis) or the code in NCBI (e.g. txid7459).

`genes` A vector of gene names (character; optional).

`speciesLevel` Whether the result should be a species-level dataset (logical).

`npar` Number of parallel searches (the default is probably the best option).

`nSearchesBatch` Number of searches per batch

Value

This function returns an object of class `data.frame` that includes the following columns. First, `Species` with the species name listed in GenBank. Second, `Ti` including the title of the accession in GenBank. Third, `Acc` listing the accession number. Fourth, `gene` including the name of the target gene.

Examples

```
## Not run:
acc.table.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"),
 speciesLevel = FALSE
)

## End(Not run)
```

`gene.sampling.retrieve` *Retrieve the distribution of genes for given organism in genbank*

Description

This function retrieves the gene names for given organism in genbank. This function is useful to explore the genes that are widely sampled for a given taxon in genbank. However, note that the performance of `gene.sampling.retrieve` is entirely dependant on the quality of submissions to genbank. For instance, if most of the sequences in genbank (nucleotide) don't have information on the sequenced region, this function won't provide reliable estimates of the sampling frequency of each gene in the database. We recommend using this function with caution, and only when there is no additional information on the genes that are widely sampled for the target group(s) of interest.

Usage

```
gene.sampling.retrieve(
  organism,
  speciesSampling = TRUE,
  npar = 2,
  nSearchesBatch = 499
)
```

Arguments

organism A vector of taxonomic groups (character).

speciesSampling Whether the results should be at the species-level (logical).

npar Number of simultaneous searches (character; optional). The default 2 is generally fast enough and does not exceed the maximum number of connections to genbank.

nSearchesBatch Number of searches per batch

Value

This function returns a data.frame that comprises the following columns. First, Gene, including the name of the relevant gene sampled in the target taxonomic groups. Second, Sampled in N species includes the number of species where the gene is sampled among the target taxa. Third, PercentOfSampledSpecies indicates the percentage of species where the gene is sampled (assuming GeneBank's taxonomic backbone).

Examples

```
## Not run:
test.spp <- gene.sampling.retrieve(organism = "Puma", speciesSampling = TRUE)
test.pop <- gene.sampling.retrieve(organism = "Puma", speciesSampling = FALSE)

## End(Not run)
```

sq.add *Add local sequences to previously downloaded sequences*

Description

This function adds sequences located in a particular folder (default is "0.AdditionalSequences") to fasta files in another folder (default is "0.Sequences"). Files must be in FASTA format and names of the files should perfectly match the ones in the previously downloaded folder (e.g. "0.Sequences"). This function creates a third new folder "0.0.OriginalDownloaded" containing the originally downloaded sequences. The sequences in the "0.Sequences" folder get replaced with the combined ones. Note that new sequences can be added even for just a fraction of the originally downloaded genes.

Usage

```
sq.add(folderDownloaded = "0.Sequences", folderNew = "0.AdditionalSequences")
```

Arguments

folderDownloaded	Name of the folder with downloaded sequences.
folderNew	Name of the folder with local sequences.

Value

None

Examples

```
## Not run:
sq.add(folderDownloaded = "0.Sequences",
       folderNew = "0.AdditionalSequences")

## End(Not run)
```

sq.aln *Align sequences*

Description

Perform multiple sequence alignment on all the fasta files saved in a given folder. Alignment is conducted using "BiocManager::install("DECIPHER")". Note that this function first optimizes the direction of the sequences, then aligns using "BDECIPHER", and finally masks the resulting alignment (optionally but does it per default). The masking step includes removing common gaps across all the species and removing highly ambiguous positions. The resulting aligned sequences are stored to a new folder "2.Alignments".'

Usage

```
sq.aln(
  folder = "1.CuratedSequences",
  FilePatterns = "renamed",
  sqs.object = NULL,
  mask = TRUE,
  maxFractionGapsSpecies = 0.01,
  ...
)
```

Arguments

`folder` Name of the folder where the sequences to align are stored (character).

`FilePatterns` A string that is common to all the target files in the relevant folder (character). Note that this argument can be set to `"NULL"` if no specific pattern wants to be analyzed.

`sqs.object` A list of sequences generated from sq.curate. Only use if you're not interested in download sequences locally.

`mask` Removes ambiguous sites (Logical, TRUE or FALSE).

`maxFractionGapsSpecies` Maximum fraction of gaps per species (when masked)

`...` Arguments passed to `"DECIPHER::AlignSeqs"`.

Value

This function will return an object of class `list` including the original and renamed sequence alignments. Optionally, this object will also include the masked alignments.

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  kingdom = "animals", folder = "0.Sequences"
)
sq.aln(folder = "1.CuratedSequences")

## End(Not run)
```

sq.curate *Curate sequences from genbank*

Description

After downloading sequences from genbank, this function curates sequences based on taxonomic information. Note that this function provides two summary datasets. First, the accession numbers. Second, the taxonomic information for each species in the database. The taxonomy strictly follows the gbif taxonomic backbone. The resulting files are saved to `"1.CuratedSequences"`. The resulting files also have the most recent curated taxonomy following the gbif (or selected database) taxonomic backbone.

Usage

```
sq.curate(
  filterTaxonomicCriteria = NULL,
  mergeGeneFiles = NULL,
  database = "gbif",
  kingdom = NULL,
  folder = "0.Sequences",
  sqs.object = NULL,
  removeOutliers = TRUE,
  minSeqs = 5,
  threshold = 0.05,
  ranks = c("kingdom", "phylum", "class", "order", "family", "genus", "species")
)
```

Arguments

`filterTaxonomicCriteria`
A single string of terms (delimited using "|") listing all the strings that could be used to identify the species that should be in the dataset (character).

`mergeGeneFiles` A named list, with each element being a character vector indicating the names of the files in `"0.Sequences"` that need to be combined into a single fasta file. For instance, you can use this argument to combine CO1 and COI.

`database` A name of a database with taxonomic information. Although 'gbif' is faster, it only has information for animals and plants. Other databases follow taxize::classification.

`kingdom` Optional and only used when database='gbif'. Two possible options: "animals" or "plants."

`folder` The name of the folder where the original sequences are located (character).

`sqs.object` A list of sequences generated from `sq.retrieve.indirect`. Only use if you're not interested in download sequences locally.

`removeOutliers` Whether `odseq:odseq` should be used to remove outliers

`minSeqs` minimum number of sequences per locus

threshold	Relative to `odseq::odseq`. Only important if `removeOutliers = TRUE`
ranks	The taxonomic ranks used to examine the taxonomy of the species in the `0.Sequences` folder.

Value

This function will return an object of class `list` with the following elements. First, the curated sequences with original names. Second, the curated sequences with species-level names. Third, the accession numbers table. Fourth, a summary of taxonomic information for all the species sampled in the files.

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  database = "gbif", kingdom = "animals",
  folder = "0.Sequences"
)

## End(Not run)
```

`sq.partitionfinderv1` *Run Partitionfinder v.1*

Description

This function runs partitionfinder v1 within phruta. For now, all analyses are based on genes. Please note that you need at least two gene regions to run partitionfinder.

Usage

```
sq.partitionfinderv1(
  folderAlignments = "2.Alignments",
  FilePatterns = "Masked",
  folderPartitionFinder = "2.1.PartitionFinderv1",
  models = "all",
  run = TRUE
)
```

Arguments

`folderAlignments`
Name of the folder where the sequences to align are stored (character).

`FilePatterns` A string that is common to all the target files in the relevant folder (character). Note that this argument can be set to `"NULL"` if no specific pattern wants to be analyzed.

`folderPartitionFinder`
Name of the new folder where the output files are stored (string).

`models` Models to run in partitionfinder (string).

`run` Run partitionfinder?

Value

None

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  kingdom = "animals", folder = "0.Sequences"
)
sq.aln(folder = "1.CuratedSequences")
sq.partitionfinderv1(
  folderAlignments = "2.Alignments",
  FilePatterns = "Masked",
  models = "all"
)

## End(Not run)
```

`sq.retrieve.direct` *Retrieve sequences from genbank*

Description

Downloads sequences from genbank (nucleotide database) for particular taxa and genes into a folder called `"0.Sequences"`.

Usage

```
sq.retrieve.direct(
  clades = NULL,
  species = NULL,
  genes = NULL,
  db = "itis",
  maxseqs = 1,
  maxlength = 5000
)
```

Arguments

`clades` A vector listing taxonomic groups of interest (character).

`species` A vector listing additional species interest (character). This argument can be used to define additional target species in the ingroup or species to be sampled in the outgroup (character).

`genes` A vector listing gene names of interest (character).

`db` Follows `db` in `taxize::downsteam`. Choose from `itis`, `gbif`, `ncbi`, `worms`, or `bold`.

`maxseqs` Maximum number of sequences to retrieve per search (taxa + gene) (numeric).

`maxlength` Maximum length of the gene sequence (numeric).

Value

None

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)

## End(Not run)
```

`sq.retrieve.indirect` *Retrieve sequences from genbank based on a dataset of accession numbers*

Description

Downloads sequences from genbank (nucleotide database) for particular taxa and genes into a folder called `"0.Sequences"`.

Usage

```
sq.retrieve.indirect(acc.table, download.sqs = FALSE)
```

Arguments

acc.table	An accession table, ideally generated using `acc.table.retrieve`. The data.frame must have the Species, Acc, and gene column names.
download.sqs	Logical indicating whether sequences should be downloaded locally or returned as a list.

Value

None

Examples

```
## Not run:
acc.table.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"),
 speciesLevel=TRUE
)

sq.retrieve.indirect(test)


## End(Not run)
```

---

tree.constraint *Tree inference under RAxML*

Description

Performs tree inference under `"RAxML"` for aligned fasta sequences in a given folder (default is `"2.Alignments"`).

Usage

```
tree.constraint(
  taxonomy_folder = "1.CuratedSequences",
  targetColumns = c("kingdom", "phylum", "class", "order", "family", "genus",
    "species_names"),
  Topology = "((ingroup), outgroup);",
  outgroup = NULL
)
```

Arguments

`taxonomy_folder`
Name of the folder where the 1.Taxonomy file is stored.

`targetColumns` Where to find "RAxML" or how to run it from the console? (string).

`Topology` A string summarizing the desired topological constraint in newick format.

`outgroup` Optional and only required when the topology argument is not "((ingroup), outgroup);".

Value

None

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  kingdom = "animals", folder = "0.Sequences"
)

tree.constraint(
  taxonomy_folder = "1.CuratedSequences",
  targetColumns = c("kingdom", "phylum", "class", "order", "family",
  "genus", "species_names"),
  Topology = "((ingroup), outgroup);",
  outgroup = "Manis_pentadactyla"
)
tree.constraint(
  taxonomy_folder = "1.CuratedSequences",
  targetColumns = c("kingdom", "phylum", "class", "order", "family",
  "genus", "species_names"),
  Topology = "((Felis), (Phoca));"
)

## End(Not run)
```

`tree.dating` *Tree dating under treePL or*

Description

Performs tree dating under `"treePL"` or `"PATHd-8"` based on secondary calibrations. Note that `"treePL"` or `"PATHd-8"` must be installed in your PATH. How to install `"PATHd-8"` in mac `"https://gist.github.com/cro`
and `"treePL"` can be installed using homebrew (brew install brewsci/bio/treepl). Thanks to Brian O'Meara and Jonathan Chang, respectively.

Usage

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

Arguments

`taxonomyFolder` Name of the folder where `"1.Taxonomy.csv"`, created duing the `"sq.curate"` step, is stored (character).

`phylogenyFolder` Name of the folder where `"RAxML_bipartitions.phruta"`, created duing the `"tree.raxml"` step, is stored (character).

`...` Arguments passed to `"geiger::congruify.phylo"`.

Value

None

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  kingdom = "animals", folder = "0.Sequences"
)
sq.aln(folder = "1.CuratedSequences")
tree.raxml(
  folder = "2.Alignments", FilePatterns = "Masked",
  raxml_exec = "raxmlHPC", Bootstrap = 100,
  outgroup = "Manis_pentadactyla"
)
tree.dating(
  taxonomyFolder = "1.CuratedSequences",
  phylogenyFolder = "3.Phylogeny", scale = "treePL"
)

## End(Not run)
```

tree.raxml *Tree inference under RAxML*

Description

Performs tree inference under "RAxML" for aligned fasta sequences in a given folder (default is "2.Alignments"). Note that you need at least two gene regions to run a partitioned analysis.

Usage

```
tree.raxml(
  folder = "2.Alignments",
  FilePatterns = "Masked_",
  raxml_exec = "raxmlHPC",
  Bootstrap = 100,
  outgroup,
  partitioned = FALSE,
  ...
)
```

Arguments

`folder` Name of the folder where the sequences to align are stored (character).

`FilePatterns` A string that is common to all the target files in the relevant folder (character). Note that this argument can be set to "NULL" if no specific pattern wants to be analized.

`raxml_exec` Where to find "RAxML" or how to run it from the console? (string).

`Bootstrap` Number of bootstrap replicates (numeric).

`outgroup` A single string of comma-separated tip labels to be used as outgroup in "RAxML" See "RAxML" documentation for more details (character).

`partitioned` Whether analyses should be partitioned by gene (Logical).

`...` Arguments passed to "ips::raxml".

Value

None

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  kingdom = "animals", folder = "0.Sequences"
)
sq.aln(folder = "1.CuratedSequences")
tree.raxml(
  folder = "2.Alignments", FilePatterns = "Masked",
  raxml_exec = "raxmlHPC", Bootstrap = 100,
  outgroup = "Manis_pentadactyla"
)

## End(Not run)
```

tree.roguetaxa *RogueNaRok within phruta*

Description

Implements the RogueNaRok algorithm for rogue taxon identification within phruta

Usage

```
tree.roguetaxa(folder = "3.Phylogeny", ...)
```

Arguments

`folder`	Name of the folder where the sequences to align are stored (character).
`...`	Arguments passed to "`Rogue::RogueTaxa`".

Value

None

Examples

```
## Not run:
sq.retrieve.direct(
  clades = c("Felis", "Vulpes", "Phoca"),
  species = "Manis_pentadactyla",
  genes = c("ADORA3", "CYTB")
)
sq.curate(
  filterTaxonomicCriteria = "Felis|Vulpes|Phoca|Manis",
  kingdom = "animals", folder = "0.Sequences"
)
sq.aln(folder = "1.CuratedSequences")
tree.raxml(
  folder = "2.Alignments", FilePatterns = "Masked",
  raxml_exec = "raxmlHPC", Bootstrap = 100,
  outgroup = "Manis_pentadactyla"
)
tree.roguetaxa(folder = "3.Phylogeny")

## End(Not run)
```

Index