Package 'phyclust'

February 10, 2021

Version 0.1-30

Date 2021-02-09

Title Phylogenetic Clustering (Phyloclustering)

Depends R (>= 3.0.0), ape

LazyLoad yes

LazyData yes

Copyright See phyclust/inst/Documents/ for files in src/msdir/, src/seq-gen/, src/paml_baseml, and R/ttzeng-*.r.

Description Phylogenetic clustering (phyloclustering) is an evolutionary Continuous Time Markov Chain model-based approach to identify population structure from molecular data without assuming linkage equilibrium. The package phyclust (Chen 2011) provides a convenient implementation of phyloclustering for DNA and SNP data, capable of clustering individuals into subpopulations and identifying molecular sequences representative of those subpopulations. It is designed in C for performance, interfaced with R for visualization, and incorporates other popular open source programs including ms (Hudson 2002) <doi:10.1093/bioinformatics/18.2.337>, seq-gen (Rambaut and Grassly 1997) <doi:10.1093/bioinformatics/13.3.235>, Hap-Clustering (Tzeng 2005) <doi:10.1002/gepi.20063> and PAML baseml (Yang 1997, 2007) <doi:10.1093/bioinformatics/13.5.555>, <doi:10.1093/molbev/msm088>, for simulating data, additional analyses, and searching the best tree. See the phyclust website for more information, documentations and examples.

BugReports https://github.com/snoweye/phyclust/issues

License GPL (>= 2)

URL https://snoweye.github.io/phyclust/

NeedsCompilation yes

Maintainer Wei-Chen Chen <wccsnow@gmail.com>

Author Wei-Chen Chen [aut, cre],
Karin Dorman [aut],
Yan-Han Chen [ctb]

Repository CRAN

Date/Publication 2021-02-10 14:00:03 UTC

R topics documented:

`phyclust-package` *Phyloclustering – Phylogenetic Clustering*

Description

This package **phyclust** (Chen 2011) implements an novel approach combining model-based clusterings and phylogenetics to classify DNA sequences and SNP sequences. Based on evolution models, sequences are assumed to follow a mutation process/distribution clouding around an unknown center ancestor. Based on Continuous Time Markov Chain Theory, mixture distributions are established to model/classify subpopulations or population structures.

The kernel part of the package are implemented in C. EM algorithms are performed to find the maximum likelihood estimators. Initialization methods for EM algorithms are also established. Several evolution models are also developed.

`ms` (Hudson 2002) and `seq-gen` (Rambaut and Grassly 1997) are two useful programs to generate coalescent trees and sequences, and both are merged into **phyclust**. `baseml` of PAML (Yang 1997, 2007) is also ported into **phyclust** and it is a program to find a phylogenetic tree by maximizing likelihood. Hap-Clustering method (Tzeng 2005) for haplotype grouping is also incorporated into **phyclust**.

Type `help(package = phyclust)` to see a list of major functions for which further documentations are available. The on-line detail instructions are also available and the link is given below in the 'References' section.

Some C and R functions and R classes of the **ape** package are also required and modified in **phyclust**.

Details

Package:	phyclust
Type:	Package
License:	GPL
LazyLoad:	yes

The main function is `phyclust` controlled by an object `.EMC` generated by a function `.EMControl`, and `find.best` can find the best solution by repeating `phyclust` with different initializations.

`ms` and `seqgen` can generate trees and sequences based on varied conditions, and they can jointly perform simulations.

`paml.baseml` can estimate trees based on sequences.

`haplo.post.prob` is a modified version of Tzeng's method for haplotype grouping which uses a evolution approach to group SNP sequences.

Some tool functions of the **ape** package are utilized in this package to perform trees in plots, check object types, and read sequence data.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Chen, W.-C. (2011) "Overlapping codon model, phylogenetic clustering, and alternative partial expectation conditional maximization algorithm", *Ph.D. Diss., Iowa Stat University*.

Hudson, R.R. (2002) "Generating Samples under a Wright-Fisher Neutral Model of Genetic Variation", *Bioinformatics*, **18**, 337-338. http://home.uchicago.edu/~rhudson1/source.html

Rambaut, A. and Grassly, N.C. (1997) "Seq-Gen: An Application for the Monte Carlo Simulation of DNA Sequence Evolution along Phylogenetic Trees", *Computer Applications In The Biosciences*, **13:3**, 235-238. http://tree.bio.ed.ac.uk/software/seqgen/

Yang, Z. (1997) "PAML: a program package for phylogenetic analysis by maximum likelihood", *Computer Applications in BioSciences*, **13**, 555-556. http://abacus.gene.ucl.ac.uk/software/paml.html

Yang, Z. (2007) "PAML 4: a program package for phylogenetic analysis by maximum likelihood", *Molecular Biology and Evolution*, **24**, 1586-1591.

Tzeng, J.Y. (2005) "Evolutionary-Based Grouping of Haplotypes in Association Analysis", *Genetics Epidemiology*, **28**, 220-231. http://www4.stat.ncsu.edu/~jytzeng/software.php

Paradis E., Claude J., and Strimmer K. (2004) "APE: analyses of phylogenetics and evolution in R language", *Bioinformatics*, **20**, 289-290. http://ape-package.ird.fr/

See Also

phyclust, .EMC, .EMControl, find.best.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

demo(package = "phyclust")
demo("ex_trees", package = "phyclust")

## End(Not run)
```

.boundary.method *Boundary Methods for Population Proportions*

Description

Methods used in EM Algorithms to deal with boundary problems of population proportions, η_k. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.boundary.method
```

Format

A character vector contains implemented boundary methods in C.

Details

The boundary value 0 of the population proportions makes the log likelihood as -Inf. Since degeneracy of subpopulations can affect the maximizing processes in EM steps. This problem is usually caused by bad initializations, and may suggest that number of cluster K may be too large.

Two methods have been implemented when any η_k less than the lower bound ($1/N$ or $1e-16$). The ADJUST (default) will adjust the η_k to the lower bound, and the IGNORE will stop the iterations and return errors.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.show.option, .init.procedure, .init.method, .EMControl, phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.boundary.method

## End(Not run)
```

.code.type *Code Types of Dataset and Substitution Models*

Description

Indicate the types of codes for datasets and functions. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.code.type
```

Format

A character vector contains implemented code types in C.

Details

Two possible types are implemented, "NUCLEOTIDE" (default) and "SNP", used in data transfers and indicating substitution models.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.show.option, .substitution.model, .EMControl, phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.code.type

## End(Not run)
```

.Color *Colors for Identifying Clusters in Plots*

Description

Color themes as used in the package **lattice**.

Usage

```
.Color
```

Format

A character vector contains colors used in plots to identify clusters.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

plotnj.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.Color

## End(Not run)
```

`.edist.model` *Evolution Distance Model*

Description

Evolution distances based on certain evolution models as in **ape**. The implemented model is used in `phyclust.edist` and initializations of EM algorithms. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.edist.model
```

Format

A character vector contains implemented evolution distances in C.

Details

This vector stores possible evolution distances implemented in **phyclust**. The default value is `D_JC69` computed form the probability of *JC69* model.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.show.option`, `phyclust.edist`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.edist.model

## End(Not run)
```

`.em.method` *EM Methods and Algorithms*

Description

The varied EM algorithms are implemented in C. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.em.method
```

Format

A character vector contains implemented EM algorithms in C.

Details

`EM` (default) stands for the standard EM algorithm, `ECM` stands for Expectation/Conditional Maximization algorithm, and `AECM` stands for Alternating ECM algorithm. The performance is roughly about `AECM` > `EM` ~ `ECM` which are dependent on the separations of data set.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Dempster, A. and Laird, N. and Rubin, D. (1977) "Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society Series B*, **39:3**, 1-38.

Meng, X.-L. and Rubin, D. (1993) "Maximum likelihood estimation via the ECM algorithm: A general framework", *Biometrika*, **80:2**, 511-567.

Meng, X.-L. and van Dyk, D. (1997) "The EM Algorithm — an Old Folk-song Sung to a Fast New Tune (with discussion)", *Journal of the Royal Statistical Society Series B*, **59**, 511-567.

See Also

`.show.option`, `.EMC`, `.EMControl`, `phyclust`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.em.method

## End(Not run)
```

.EMC *EM Control*

Description

An default template object stores controlling options for phyclust to perform EM algorithms. This object combines all other read-only objects and more required options for EM algorithms. This is essential for phyclust and other related functions.

Usage

```
.EMC
```

Format

A list contains all controlling options

Details

A list created by .EMControl contains all controlling options for EM algorithms. This list will be directly passed to C codes and control the every things of EM algorithms.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.show.option, .EMControl, phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.EMC

## End(Not run)
```

`.EMControl` *EM Control Generator*

Description

Generate an EM control (`.EMC`) controlling the options, methods, conditions and models of EM algorithms. As `.EMC`, this function generate a default template. One can either modify `.EMC` or employ this function to control EM algorithms.

Usage

```
.EMControl(exhaust.iter = 1, fixed.iter = 5,
    short.iter = 100, EM.iter = 1000,
    short.eps = 1e-2, EM.eps = 1e-6,
    cm.reltol = 1e-8, cm.maxit = 5000,
    nm.abstol.Mu.given.QA = 1e-8, nm.reltol.Mu.given.QA = 1e-8,
    nm.maxit.Mu.given.QA = 500,
    nm.abstol.QA.given.Mu = 1e-8, nm.reltol.QA.given.Mu = 1e-8,
    nm.maxit.QA.given.Mu = 5000,
    est.non.seg.site = FALSE, max.init.iter = 50,
    init.procedure = .init.procedure[1],
    init.method = .init.method[1],
    substitution.model = .substitution.model$model[1],
    edist.model = .edist.model[1], identifier = .identifier[1],
    code.type = .code.type[1], em.method = .em.method[1],
    boundary.method = .boundary.method[1], min.n.class = 1,
    se.type = FALSE, se.model = .se.model[1], se.constant = 1e-2)
```

Arguments

`exhaust.iter`	number of iterations for "exhaustEM", default = 1.
`fixed.iter`	number of iterations for "RndpEM", default = 5.
`short.iter`	number of short-EM steps, default = 100.
`EM.iter`	number of long-EM steps, default = 1000.
`short.eps`	tolerance of short-EM steps, default = 1e-2.
`EM.eps`	tolerance of long-EM steps, default = 1e-6.
`cm.reltol`	relative tolerance for a CM step, default = 1e-8
`cm.maxit`	maximum number iteration for a CM step, default = 5000.
`nm.abstol.Mu.given.QA`	see 'Details', default = 1e-8
`nm.reltol.Mu.given.QA`	see 'Details', default = 1e-8
`nm.maxit.Mu.given.QA`	see 'Details', default = 500.

`nm.abstol.QA.given.Mu`
: see 'Details', default = 1e-8

`nm.reltol.QA.given.Mu`
: see 'Details', default = 1e-8

`nm.maxit.QA.given.Mu`
: see 'Details', default = 5000.

`est.non.seg.site`
: estimate non-segregation sites, default = FALSE.

`max.init.iter`
: maximum number of initialization iteration, default = 50.

`init.procedure`
: initialization procedure, default = "exhaustEM".

`init.method`
: initialization method, default = "randomMu".

`substitution.model`
: substitution model, default = "JC69".

`edist.model`
: evolution distance, default = "D_J69".

`identifier`
: identifier, default = "EE".

`code.type`
: code type, default = "NUCLEOTIDE".

`em.method`
: EM method, default = "EM".

`boundary.method`
: boundary method, default = `ADJUST`.

`min.n.class`
: minimum number of sequences in a cluster, default = 1.

`se.type`
: sequencing error type, default = FALSE.

`se.model`
: sequencing error model, default = "CONVOLUTION".

`se.constant`
: constrained constant, default = 1e-2.

Details

`exhaust.iter`, `fixed.iter`, `short.iter`, and `short.eps` are used to control the iterations of initialization procedures and methods.

`EM.iter` and `EM.eps` are used to control the EM iterations.

`cm.reltol` and `cm.maxit` are used to control the ECM iterations.

Arguments starting with `nm.` are options for the Nelder-Mead method as in `optim`. The C codes of Nelder-Mead are modified from the R math library and the options are all followed. `abstol` and `reltol` are for absolute and relative tolerances. `Mu.given.QA` is for maximizing the profile function of μ_k given Q_k, and `QA.given.Mu` is for maximizing the profile function of Q_k given μ_k.

`est.non.seg.site` indicates whether to estimate the states of center sequences. If FALSE, the states will be fixed as the non segregating sites. Usually, there is no need to estimate.

`max.init.iter` is for certain initialization methods, e.g. `randomNJ` and `K-Medoids` need few tries to obtain an appropriate initial state.

`init.procedure` and `init.method` are for initializations.

`min.n.class` is the minimum number of sequences in a cluster to avoid bad initialization state and degenerated clusters.

`se.type`, `se.model`, and `se.constant` which are used only for sequencing error models and only for nucleotide data without labels.

Value

This function returns a list as `.EMC`.

The sequencing error controls are stored in `se.type`, `se.model`, and `se.constant`, for sequencing error type, model, and constrained constant of errors, respectively.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.show.option`, `.EMC`, `.boundary.method`, `.code.type`, `.edist.model`, `.em.method`, `.identifier`, `.init.method`, `.init.procedure`, `.substitution.model`, `optim`, `phyclust`, `phyclust.se`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# The same as .EMC
.EMControl()

# Except code.type, all others are the same as .EMC
.EMControl(code.type = "SNP")
.EMControl(code.type = .code.type[2])

## End(Not run)
```

.identifier *Identifiers for Evolution Models*

Description

Identifiers for evolution models identify the Q_k matrix and evolution time t_k for subpopulations. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.identifier
```

Format

A character vector contains implemented identifiers in C.

Details

Four major identifiers are implemented in C, `EE`, `EV`, `VE`, and `VV`. The first letter indicates the structure for Q_k matrix, and the second letter indicates the evolution time t_k for subpopulations. `E` and `V` indicate equal and varied for all subpopulations.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.show.option`, `.EMC`, `.EMControl`, `phyclust`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.identifier

## End(Not run)
```

.init.method *Initialization Methods for EM Algorithms*

Description

The varied initialization methods are implemented in C. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.init.method
```

Format

A character vector contains implemented initialization methods in C.

Details

randomMu, NJ, randomNJ, PAM, K-Medoids and manualMu are implemented where the codes for the NJ are modified from **ape**, and the codes for the PAM method are modified from **cluster**. These methods are only provide initializations for EM algorithms.

- 'randomMu'randomly picks centers and assigns all sequences near by the center according an evolution distance.
- 'NJ'bases on a neighbor-joining tree and partitions the tree by the long branches into subtrees to form clusters.
- 'randomNJ'randomly partition a neighbor-joining tree into subtrees to form clusters.
- 'PAM'uses the partition around medoids algorithm to locate the centers of dataset.
- 'K-Medoids'performs K-Means types algorithms to randomly and roughly locate centers and form clusters.
- 'manualMu'requires a vector containing class ids for all sequences.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Saitou, N. and Nei, M. (1987), "The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees", *Molecular Biology and Evolution*, **4:4**, 406-425.

Kaufman, L. and Rousseeuw, P.J. (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley.

Theodoridis, S. and Koutroumbas, K. (2006), *Pattern Recognition*, 3rd ed., pp. 635.

See Also

.show.option, .init.procedure, .EMControl, phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.init.method

## End(Not run)
```

`.init.procedure` *Initialization Procedures for EM Algorithms*

Description

The varied initialization procedures are implemented in C. The first element is the default value. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.init.procedure
```

Format

A character vector contains implemented initialization procedures in C.

Details

`exhaustEM`, `emEM`, `RndEM`, and `RndpEM` are implemented. Based on initialization states given by a initialization method, see `.init.method` for more information. These procedures will search a better starting states for final EM steps.

- 'exhaustEM'runs each initialization with EM steps until convergent, and pick the best one of the convergence as the return result.
- 'emEM'uses few short EM steps to improve initialization, then pick the best of initialization state for long EM steps, and returns the final result.
- 'RandEM'bases on initialization methods to generate initialization states, the number is equal to short EM steps, then pick the best of initialization state for long EM steps, and returns the final result.
- 'RandEM'bases on initialization methods to generate initialization states and run a fixed number of EM steps, until total steps exhaust short EM steps, then pick the best of initialization state for long EM steps, and returns the final result.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Biernacki, C. and Celeux, G. and Govaert, G. (2003) "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models", *Computational Statistics and Data Analysis*, **41:3**, 561-575.

Maitra, R. (2009) "Initializing partition-optimization algorithms", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **6:1**, 114-157.

See Also

.show.option, .init.method, .EMControl, phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.init.procedure

## End(Not run)
```

.label.method *Label Method*

Description

An object stores label method for un-, semi-, and general semi- supervised clustering.. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.label.method
```

Format

A character vector contains implemented evolution distances in C.

Details

This vector stores possible label methods implemented in **phyclust**. The default value is NONE for unsupervised clustering. SEMI is for semi-supervised clustering, and GENERAL is for general semi-supervised clustering. Only un- and semi-supervised clustering are implemented.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.label.method

## End(Not run)
```

`.se.model` *Sequencing Error Model*

Description

An object stores sequencing error models.

Usage

```
.se.model
```

Format

A character vector contains all possible sequencing models.

Details

Currently, only a `CONVOLUTION` model is implemented.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.show.option`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.se.model

## End(Not run)
```

`.show.option` *Show Available Options*

Description

This function show available options for functions in **phyclust**.

Usage

```
.show.option()
```

Details

This function show some available options for functions in **phyclust**. They are used in `.EMControl`, `phyclust`, ... etc, and options are stored in several objects separately. They will be passed into C, so the elemental order are important. Basically, they are all read-only objects.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.boundary.method`, `.code.type`, `.edist.model`, `.em.method`, `.EMC`, `.EMControl`, `.identifier`, `.init.method`, `.init.procedure`, `.nucleotide`, `.snp`, `.substitution.model`,

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.show.option()

## End(Not run)
```

`.substitution.model` *Substitution Models for Mutation Processes*

Description

An object stores substitution models for mutation processes for Continuous Time Markov Chain theory. **This is a read-only object and the elemental order is followed in C.**

Usage

```
.substitution.model
```

Format

A data frame contains two character vectors, `mode` and `code.type`.

Details

This data frame indicates substitution models implemented in C.

- 'model': names of substitution models for mutations.
- 'code.type': code types of substitution models, either NUCLEOTIDE or SNP.

The major models are:

Model	Author and Publication	Parameter
JC69	Jukes and Cantor 1969.	t
K80	Kimura 1980.	κ, t
F81	Felsenstein 1981.	π, t
HKY85	Hasegawa, Kishino, and Yano 1985.	π, κ, t

Other models starting with `E_` use empirical frequencies for equilibrium probabilities.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Jukes, T. H. and Cantor, C. R. (1969) "Evolution of Protein Molecules", *Mammalian Protein Metabolism*, **3**, 21-132

Kimura, M. (1980) "A Simple Method for Estimating Evolutionary Rates of Base Substitutions through Comparative Studies of Nucleotide Sequences", *Journal of Molecular Evolution*, **16**, 111-120

Felsenstein, J. (1981) "Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach", *Journal of Molecular Evolution*, **17**, 368-376

Hasegawa, M. and Kishino, H. and Yano, T. (1985) "Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA", *Journal of Molecular Evolution*, **22:2**, 160-174

See Also

`.show.option`, `.code.type`, `.identifier`, `.EMControl`, `phyclust`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.substitution.model

## End(Not run)
```

`as.star.tree` *Coerce a Rooted Tree to a Star Tree in Class phylo*

Description

Coerce a rooted tree generating by `ms` to a star tree and maintain a bifurcation structure.

Usage

```
as.star.tree(rooted.tree, keep.bifurcation = TRUE)
```

Arguments

`rooted.tree`	a rooted tree in `Class phylo`.
`keep.bifurcation`	keep a bifurcation structure.

Details

A tree with a star shape means that all internal branches are 0 and all leaf branches are equal.

The `rooted.tree` should be in a `phylo` class of **ape**, and may be created by `ms`.

Basically, it is a list with an attribute that the class is phylo, and the other elements are:

- 'edge'edge ids.
- 'Nnode'number of internal nodes.
- 'tip.lab'number of tips (leaves).
- 'edge.length'length of edges.

If `keep.bifurcation` is TRUE, then internal branches are set to be 0 and leaves branches are set to the original tree height. Otherwise, the internal branches will be dropped from rooted.tree.

Value

Return a rooted tree in Class phylo with a star shape.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

ms, read.tree, as.phylo, plot.phylo.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
ret.ms <- ms(5, 1, opts = paste("-T", sep = " "))
tree.ms <- read.tree(text = ret.ms[3])
str(tree.ms)
(tree.star <- as.star.tree(tree.ms))

# Plot results
par(mfrow = c(1, 2))
plot(tree.ms, type = "u", main = "original tree")
plot(tree.star, type = "u", main = "as star tree")

## End(Not run)
```

---

bootstrap.seq *Bootstrap Sequences from a Fitted Model and Star Tree.*

Description

This function bootstraps sequences from a model fitted by phyclust and star trees generated by bootstrap.star.trees. The fitted model can be varied in .identifier.

Usage

```
bootstrap.seq(ret.phyclust, star.trees)
```

Arguments

ret.phyclust	a phyclust object in Class phyclust.
star.trees	star trees might be generated by bootstrap.star.trees.

Details

ret.phyclust is a phyclust object in Class phyclust which is usually fitted by phyclust, or returned by phyclust.m.step.

star.trees should be corresponding to the ret.phyclust which might be directly bootstrapped from the function bootstrap.star.trees.

Value

Return a list containing sequences in K clusters.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, bootstrap.star.trees, bootstrap.star.trees.seq.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)

ret.1 <- phyclust(seq.data.toy$org, 2, EMC = EMC.1)
ret.tree <- bootstrap.star.trees(ret.1)
ret.seq <- bootstrap.seq(ret.1, ret.tree)

## End(Not run)
```

---

bootstrap.seq.data *Bootstrap a seq.data from a Fitted Model.*

Description

This function simplifies the bootstrap function bootstrap.star.trees.seq(), and only return a list object with class seq.data.

Usage

```
bootstrap.seq.data(ret.phyclust, min.n.class = 1)
```

Arguments

`ret.phyclust`	a phyclust object in `Class` `phyclust`.
`min.n.class`	minimum number of sequences for a cluster.

Details

`ret.phyclust` is a phyclust object in `Class` `phyclust` which is usually fitted by `phyclust`, or returned by `phyclust.m.step`.

`min.n.class` is a boundary condition to avoid degenerate clusters when some population proportions, η_k, are small in the fitted model.

Value

Return an object in `Class` `seq.data` as the result from `read.*()`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`phyclust`, `bootstrap.star.trees`, `Class` `seq.data`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)

ret.1 <- phyclust(seq.data.toy$org, 2, EMC = EMC.1)
(ret.all <- bootstrap.seq.data(ret.1))

## End(Not run)
```

`bootstrap.star.trees` *Bootstrap a Star Tree from a Fitted Model.*

Description

This function bootstraps a star tree from a model fitted by `phyclust`. Each cluster corresponds to a star tree and a center sequence where sequences will evolve from. This function is called by `bootstrap.star.trees.seq` to generate sequences. The fitted model can be varied in `.identifier`.

Usage

```
bootstrap.star.trees(ret.phyclust, min.n.class = 1)
```

Arguments

`ret.phyclust`	a phyclust object in `Class phyclust`.
`min.n.class`	minimum number of sequences for a cluster.

Details

`ret.phyclust` is a phyclust object in `Class phyclust` which is usually fitted by `phyclust`, or returned by `phyclust.m.step`.

`min.n.class` is a boundary condition to avoid degenerate clusters when some population proportions, η_k, are small in the fitted model.

Value

Return a list containing K star trees according to `ret.phyclust`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`phyclust`, `bootstrap.seq`, `bootstrap.star.trees.seq`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)

ret.1 <- phyclust(seq.data.toy$org, 2, EMC = EMC.1)
ret.trees <- bootstrap.star.trees(ret.1)

## End(Not run)
```

`bootstrap.star.trees.seq`

Bootstrap Sequences from a Fitted Model.

Description

This function bootstraps sequences from a model fitted by `phyclust` by combining two functions `bootstrap.star.trees` and `bootstrap.seq`. The fitted model can be varied in `.identifier`.

Usage

```
bootstrap.star.trees.seq(ret.phyclust, min.n.class = 1)
```

Arguments

`ret.phyclust`	a phyclust object in `Class phyclust`.
`min.n.class`	minimum number of sequences for a cluster.

Details

`ret.phyclust` is a phyclust object in `Class phyclust` which is usually fitted by `phyclust`, or returned by `phyclust.m.step`.

`min.n.class` is a boundary condition to avoid degenerate clusters when some population proportions, η_k, are small in the fitted model.

Value

Return a list containing two elements, and both are corresponding to the model of `ret.phyclust`, including:

`trees`	a list, K star trees according to `ret.phyclust`
`seq`	a list, sequences in K clusters

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, bootstrap.star.trees, bootstrap.seq.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)

ret.1 <- phyclust(seq.data.toy$org, 2, EMC = EMC.1)
ret.all <- bootstrap.star.trees.seq(ret.1)

## End(Not run)
```

---

code2nid *Transfer Codes (A, G, C, T, -) and nids (0, 1, 2, 3, 4)*

Description

Transfer nucleotide codes (A, G, C, T, -) and nucleotide ids (0, 1, 2, 3, 4).

Usage

```
### S3 methods for a list, vector or matrix (default).
code2nid(codeseq)
nid2code(nidseq, lower.case = TRUE)
```

Arguments

codeseq	a character vector contains nucleotide codes, A, G, C, T, or -.
nidseq	a numerical vector contains nucleotide ids, 0, 1, 2, 3, or 4.
lower.case	transfer in lower cases.

Details

These functions are based on the internal object .nucleotide to transfer codes and nids.

Value

code2nid returns a numerical vector containing nucleotide ids, and nid2code returns a character vector containing nucleotide codes.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.nucleotide, snp2sid, sid2snp, code2snp, snp2code.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

a <- c("A", "C", "G", "-", "T")
code2nid(a)
nid2code(code2nid(a))

## End(Not run)
```

---

code2snp *Transfer Nucleotide Codes / nids and SNPs / sids*

Description

Transfer nucleotide codes (A, G, C, T, -) and SNPs (1, 2, -). Transfer nucleotide ids (0, 1, 2, 3, 4) and SNP ids (0, 1, 2).

Usage

```
### S3 methods for a list, vector or matrix (default).
code2snp(codeseq)
snp2code(snpseq, half = TRUE)

nid2sid(nidseq)
sid2nid(sidseq, half = TRUE)
```

Arguments

codeseq	a character vector contains nucleotide codes, A, G, C, T, or -.
snpseq	a character vector contains SNPs, 1, 2, or -.
half	nucleotide codes will be half assigned, see the 'Details' for more information.
nidseq	a numerical vector contains nucleotide ids, 0, 1, 2, 3, or 4.
sidseq	a numerical vector contains SNP ids, 0, 1, or 2.

Details

These functions are based on the internal object .nucleotide and .snp to transfer nucleotide codes and SNPs. For code2snp, A, G are transfered to 1, and C, T are transfered to 2. For snp2code, 1 is transfered half to A and G, and 2 is transfered half to C and T if half = TRUE. Otherwise, 1 is all transfered to A, and 2 is all transfered to C.

Value

code2nid returns a character vector containing nucleotide ids, and nid2code returns a character vector containing nucleotide codes.

nid2sid returns a numerical vector containing SNP ids, and sid2nid returns a numerical vector containing nucleotide ids.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.nucleotide, .snp, code2nid, nid2code, snp2sid, sid2snp.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# For codes
a.vector <- c("A", "C", "G", "-", "T")
code2snp(a.vector)
snp2code(code2snp(a.vector))
snp2code(code2snp(a.vector), half = FALSE)

# For ids
a.sid.vector <- c(0, 2, 1, 4, 3)
nid2sid(a.sid.vector)
sid2nid(nid2sid(a.sid.vector))
sid2nid(nid2sid(a.sid.vector), half = FALSE)
```

```
# Test list
a.list <- list(a, a)
code2snp(a.list)
snp2code(code2snp(a.list))
snp2code(code2snp(a.list), half = FALSE)

# Test matrix
a.matrix <- rbind(a, a)
code2snp(a.matrix)
snp2code(code2snp(a.matrix))
snp2code(code2snp(a.matrix), half = FALSE)

## End(Not run)
```

`data.fasta.pony` *Great Pony 625 EIAV rev Dataset in the Fasta Format*

Description

Great pony 625 EIAV dataset is published by Baccam, P., et al. (2003), and they are also available on NCBI database. This is a follow-up study of `Data Pony 618`.

Format

A text file in fasta format is stored in the data subdirectory.

Details

EIAV rev dataset contains 62 nucleotide sequences and 406 sites.

Author(s)

Baccam, P., et al. (2003).

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Baccam, P., et al. (2003) "Subpopulations of Equine Infectious Anemia Virus Rev Coexist In Vivo and Differ in Phenotype", *Journal of Virology*, **77**, 12122-12131.

See Also

`read.phylip`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

data.path <- paste(.libPaths()[1], "/phyclust/data/pony625.fas", sep = "")
# edit(file = data.path)
my.pony.625 <- read.fasta(data.path)
str(my.pony.625)

## End(Not run)
```

---

data.phylip.crohn *Crohn's Disease SNP Dataset in the phylip Format*

Description

Crohn's disease dataset is published by Hugot, et al. (2001).

Format

A text file in phylip format is stored in the data subdirectory.

Details

Crohn's disease dataset is used to perform haplotype grouping used in Tzeng's paper (2005).

Totally, 1102 haplotypes/SNP sequences and 8 sites.

Author(s)

Hugot, J.P., et al. (2001).

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Hugot, J.P., et al. (2001) "Association of NOD2 Leucine-Rich Repeat Variants with Susceptibility to Crohn's Disease", *Nature*, **411**, 599-603.

Tzeng, J.Y. (2005) "Evolutionary-Based Grouping of Haplotypes in Association Analysis", *Genetics Epidemiology*, **28**, 220-231. http://www4.stat.ncsu.edu/~jytzeng/software.php

See Also

read.phylip.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

data.path <- paste(.libPaths()[1], "/phyclust/data/crohn.phy", sep = "")
# edit(file = data.path)
my.snp <- read.phylip(data.path, code.type = "SNP")
str(my.snp)

## End(Not run)
```

`data.phylip.pony` *Great Pony 524 EIAV rev Dataset in the phylip Format*

Description

Great pony 524 EIAV dataset is published by Baccam, P., et al. (2003), and they are also available on NCBI database. There is a follow-up study, `Data Pony 625`.

Format

A text file in phylip format is stored in the data subdirectory.

Details

EIAV rev dataset contains 146 nucleotide sequences and 405 sites.

Author(s)

Belshan, M., et al. (2001).

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Belshan, M., et al. (2001) "Genetic and Biological Variation in Equine Infectious Anemia Virus Rev Correlates with Variable Stages of Clinical Disease in an Experimentally Infected Pony", *Virology*, **279**, 185-200.

See Also

`read.fasta`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

data.path <- paste(.libPaths()[1], "/phyclust/data/pony524.phy", sep = "")
# edit(file = data.path)
my.pony.524 <- read.phylip(data.path)
str(my.pony.524)

## End(Not run)
```

`file.read` *Read Data from Files by Formats and Return a seq.data Object*

Description

Two major file formats are supported in **phyclust**, `Format phylip` and `Format fasta`. **These functions only read files in basic syntax**, and return an object in `Class seq.data`.

Usage

```
read.fasta(filename, byrow = TRUE, code.type = .code.type[1], aligned = TRUE,
           sep = "")
read.fasta.format(filename, byrow = TRUE, aligned = TRUE, sep = "")

read.phylip(filename, byrow = TRUE, code.type = .code.type[1], sep = "")
read.phylip.format(filename, byrow = TRUE, sep = "")
```

Arguments

`filename`	a file name where data is read from.
`byrow`	advanced option, default = TRUE.
`code.type`	either "NUCLEOTIDE" (default) or "SNP".
`aligned`	indicate aligned data.
`sep`	use to split sites, "" (default) and "," for "CODON".

Details

For unaligned sequences, `read.fasta` returns a list storing data. `read.phylip` is only for aligned data and returns a matrix.

`read.fasta.format` and `read.phylip.format` will read in original coding without any transformation as `code.type = NULL` in `write.fasta` and `write.phylip`. Suppose these functions return an object `ret`, one can write other functions `ret2aa()` to post transform the coding and replace `ret$org` by the results of `ret2aa(ret$org.code)`.

`byrow` indicates the data will be store by row or not. Usually, the default is TRUE. The FALSE is only for advance users with careful manipulations and for speeding up the bootstraps.

`sep` can specify a character which is used to split sites in file. By default, "" denote no character between sites. Only "CODON" id requires a character to avoid ambiguity

Value

Return an object in Class seq.data.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

write.fasta, write.phylip.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# PHYLIP
data.path <- paste(.libPaths()[1], "/phyclust/data/crohn.phy", sep = "")
(my.snp <- read.phylip(data.path, code.type = "SNP"))

# FASTA
data.path <- paste(.libPaths()[1], "/phyclust/data/pony625.fas", sep = "")
(my.pony <- read.fasta(data.path))

## End(Not run)
```

---

file.write *Write Data to Files by Formats*

Description

Two major file formats are supported in **phyclust**, Format phylip and Format fasta. **These functions only write files in basic syntax.**

Usage

```
write.fasta(seqdata, filename, classid = NULL, seqname = NULL,
    width.line = 60, lower.case = FALSE, code.type = .code.type[1],
    sep = "")
write.fasta.format(seqdata, filename, classid = NULL, seqname = NULL,
    width.line = 60, sep = "")

write.phylip(seqdata, filename, classid = NULL, seqname = NULL,
    width.seqname = 10, width.line = 60, lower.case = FALSE,
```

```
    code.type = .code.type[1], sep = "")

write.phylip.format(seqdata, filename, classid = NULL, seqname = NULL,
    width.seqname = 10, width.line = 60, sep = "")


write.paml(seqdata, filename, classid = NULL, seqname = NULL,
    width.seqname = 10, width.line = 60, lower.case = FALSE,
    code.type = .code.type[1], sep = "")
write.paml.format(seqdata, filename, classid = NULL, seqname = NULL,
    width.seqname = 10, width.line = 60, sep = "")
```

Arguments

seqdata	a matrix contains sequence ids as X in phyclust.
filename	a file name where data is written to.
classid	class id of sequences.
seqname	sequence names.
width.seqname	number of characters of sequence names to be stored.
width.line	width of lines for breaking lines.
lower.case	use lower case of letters to write
code.type	either "NUCLEOTIDE" (default) or "SNP".
sep	a character to split sites, "" (default) and "," for "CODON".

Details

write.fasta, write.phylip, and write.paml are general functions call write.fasta.format, write.phylip.format and write.paml.format.

write.fasta.format, write.phylip.format, and wirte.paml.format will not do any transformation for input sequences, but directly write them into the file as code.type = NULL in write.fasta, write.phylip and write.paml.

Note that PAML uses one of PHYLIP format to deal with sequence files, so write.paml.format is to write files in a different format of write.phylip.format. The main purpose of write.paml and write.paml.format is to generate files for pamle.baseml.

sep can specify a character which is used to split sites in file. By default, "" denote no character between sites. Only "CODON" id requires a character to avoid ambiguity.

Value

Save a text file.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

read.fasta, read.phylip.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# PHYLIP
data.path <- paste(.libPaths()[1], "/phyclust/data/crohn.phy", sep = "")
my.snp <- read.phylip(data.path, code.type = "SNP")
write.phylip(my.snp$org, "new.crohn.phy", code.type = "SNP")

# FASTA
data.path <- paste(.libPaths()[1], "/phyclust/data/pony625.fas", sep = "")
(my.pony <- read.fasta(data.path))
write.fasta(my.pony$org, "new.pony.fas")

# PAML
write.paml(my.pony$org, "new.pony.pam")

# Amino acid in PHYLIP
aa.aid <- nid2aid(my.pony$org)
aa.acode <- aid2acode(aa.aid)
write.phylip(aa.aid, "new.pony.aa.phy", code.type = "AMINO_ACID")
write.phylip.format(aa.aid, "new.pony.aa.aid.phy", sep = ",")
write.phylip.format(aa.acode, "new.pony.aa.acode.phy")

# Amino acid in FASTA
write.fasta(aa.aid, "new.pony.aa.phy", code.type = "AMINO_ACID")
write.fasta.format(aa.aid, "new.pony.aa.aid.phy", sep = ",")
write.fasta.format(aa.acode, "new.pony.aa.acode.fas")

# Amino acid in PAML
write.paml(aa.aid, "new.pony.aa.pam", code.type = "AMINO_ACID")
write.paml.format(aa.aid, "new.pony.aa.aid.pam", sep = ",")
write.paml.format(aa.acode, "new.pony.aa.acode.pam")

## End(Not run)
```

---

find.best *Find the Best Solution of phyclust*

Description

Based on input initialization procedures and methods, this function tries to find the best solution in terms of the highest log-likelihood value.

Usage

```
find.best(X, K, EMC = .EMC, manual.id = NULL, byrow = TRUE,
    init.procedure = .init.procedure, init.method = .init.method,
    file.tmp = NULL, visible = FALSE, save.all = FALSE)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
K	number of clusters.
EMC	EM control.
manual.id	manually input class ids.
byrow	advanced option for X, default = TRUE.
init.procedure	customized initialization procedures.
init.method	customized initialization methods.
file.tmp	a file for saving temporary results.
visible	TRUE for reporting iterations.
save.all	TRUE for saving all results.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Note: gaps - are not supported yet, drop them from data.

EMC contains all options used for EM algorithms.

manual.id manually input class ids as an initialization only for the initialization method, 'manualMu'.

byrow used in bootstraps to avoid transposing matrix 'X'. If FALSE, then the 'X' should be have the dimension $L \times K$.

init.procedure and init.method are methods for searching the best result. This function will try all combinations of these two options.

file.tmp is used to save temporary results due to long computing. If NULL, there will no saving in each combinations.

Value

An list with class phyclust will be returned containing several elements, see phyclust for detail.

ToDo(s)

- implement codes for gaps -.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.EMC, .EMControl. phyclust.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMControl(exhaust.iter = 1, short.iter = 5, EM.iter = 5)
(ret.1 <- find.best(seq.data.toy$org, 2, EMC = EMC.1))

## End(Not run)
```

---

find.consensus *Find the Consensus Sequence*

Description

Based on the input data, this function will search all data along all sites to find a consensus sequence which may be or may not be one of the data.

Usage

```
find.consensus(X, code.type = .code.type[1], with.gap = FALSE)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
code.type	either "NUCLEOTIDE" (default) or "SNP".
with.gap	FALSE (default) for no gap in consensus sequence.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Value

A vector containing the consensus sequence with length L will be returned.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

plotdots.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

find.consensus(seq.data.toy$org)

## End(Not run)
```

`gen.equal.star.anc.dec` *Generate Comprehensive Trees.*

Description

Generate comprehensive trees for simulation studies.

Usage

```
gen.equal.star.anc.dec(K, N.K, rate.f = 0.5)
```

Arguments

`K`	number of clusters, K.
`N.K`	number of sequences for each cluster, a vector with length `K`.
`rate.f`	r_f, growth rate ratio of ancestral and descendent trees.

Details

These functions generates an ancestral tree in K tips and generates descendent trees according to `N.K` tips. All trees, ancestral and descendent, are coerced to star shapes and scaled their heights to fit the ratio `rate.f`, and the final tree has total height 1. The returns are stored in a list, and the final tree is stored with a name `equal.star`.

Value

A list contains all information of generation and results including:

'K'	number of clusters.
'N.K	number of sequences for each cluster.
'rate.f'	r_f, growth rate ratio of ancestral and descendent trees.
'anc'	an ancestral tree.
'dec'	all descendent trees.
'equalstar'	a tree that descendants are equal star trees.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

gen.unit.K.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
tree.K <- gen.equal.star.anc.dec(6, rep(3:5, 2),
                                 rate.f = 0.7)
X.class <- as.numeric(gsub("d(.).(.)", "\\1",
                           tree.K$equal.star$tip.label))

# Plot results
plotnj(tree.K$equal.star, X.class, type = "p",
       edge.width.class = 2, main = "equal.star")
axis(1)

## End(Not run)
```

gen.seq *Generate Sequences Given a Rooted Tree.*

Description

These functions call seqgen to generate sequences by evolutions models based on a rooted tree. gen.seq.HKY is to generate nucleotide sequences, and gen.seq.SNP is to generate SNP sequences.

Usage

```
gen.seq.HKY(rooted.tree, pi, kappa, L, rate.scale = 1,
            anc.seq = NULL)
gen.seq.SNP(rooted.tree, pi, L, rate.scale = 1, anc.seq = NULL)
```

Arguments

rooted.tree	a rooted tree in Class phylo.
pi	equilibrium probabilities, sums to 1.
kappa	transition and transversion bias.
L	number of sites.
rate.scale	a scale to all branch lengths.
anc.seq	an ancestral sequence either in nids or sids, length = L.

Details

The rooted.tree should be in a phylo class of **ape**, and may be created by ms.

The pi has length 4 for nucleotide sequences, and 2 for SNP sequences.

The rate.scale is equivalent to rescale the height of rooted.tree.

Value

Return an object in Class seqgen.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

gen.star.tree, seqgen.

Examples

```
library(phyclust, quiet = TRUE)

# Generate a tree
set.seed(1234)
ret.ms <- ms(nsam = 5, nreps = 1, opts = "-T")
tree.ms <- read.tree(text = ret.ms[3])

# Generate nucleotide sequences
anc.HKY <- rep(0:3, 3)
pi.HKY <- c(0.2, 0.2, 0.3, 0.3)
kappa <- 1.1
L <- length(anc.HKY)
set.seed(1234)
paste(nid2code(anc.HKY, lower.case = FALSE), collapse = "")
(HKY.1 <- gen.seq.HKY(tree.ms, pi.HKY, kappa, L, anc.seq = anc.HKY))

# evolve 5 times longer
(HKY.2 <- gen.seq.HKY(tree.ms, pi.HKY, kappa, L,
                      rate.scale = 5, anc.seq = anc.HKY))

# Generate SNP sequences
anc.SNP <- rep(0:1, 6)
pi.SNP <- c(0.4, 0.6)
L <- length(anc.SNP)
set.seed(1234)
paste(sid2snp(anc.SNP), collapse = "")
(SNP.1 <- gen.seq.SNP(tree.ms, pi.SNP, L, anc.seq = anc.SNP))

# evolve 5 times longer
(SNP.2 <- gen.seq.SNP(tree.ms, pi.SNP, L, rate.scale = 5,
                      anc.seq = anc.SNP))
```

`gen.star.tree` *Generate a Rooted Tree with a Star Shape*

Description

Generate a rooted tree with a star shape based on a sequence calls of several functions.

Usage

```
gen.star.tree(N, total.height = 1)
```

Arguments

`N`	number of leaves.
`total.height`	total tree height.

Details

A tree with a star shape means that all internal branches are 0 and all leaf branches are equal.

This function combining with `gen.seq.HKY` or `gen.seq.SNP` is used in simulation studies and bootstrap tree samples

Value

Return a rooted tree in `Class` `phylo` with a star shape.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`ms`. `as.star.tree`, `get.rooted.tree.height`. `rescale.rooted.tree`. `as.phylo`, `plot.phylo`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

ret.star <- gen.star.tree(5)
plot(ret.star, type = "u")

## End(Not run)
```

---

gen.unit.K *Generate Comprehensive Trees.*

Description

Generate comprehensive trees for simulation studies.

Usage

```
gen.unit.K(K, N.K, rate.anc = 10, rate.dec = 10)
```

Arguments

`K`	number of clusters, K.
`N.K`	number of sequences for each cluster, a vector with length `K`.
`rate.anc`	r_a, growth rate of ancestral tree.
`rate.dec`	r_d, growth rate of descendent tree.

Details

These functions generates an ancestral tree in K tips and generates descendent trees according to `N.K` tips, and returns several types of trees, `org`, `equal`, `max`, and `star`, as the following:

- 'org': original tree, adjacent the descendent trees to the ancestral tree.
- 'equal': descendent trees are scaled to the average height and attached to the ancestral tree, then scale the total height to be 1.
- 'max': descendent trees are attached to the ancestral tree, then scale the maximum height to be 1.
- 'star': descendent trees are applied `as.star.tree` and attached to the ancestral tree, then scale the maximum height to be 1.

Value

A list contains all information of generation and results including:

`'K'`	number of clusters.
`'N.K`	number of sequences for each cluster.
`'rate.anc'`	r_a, growth rate of ancestral tree.
`'rate.dec'`	r_d, growth rate of descendent tree.
`'height.anc'`	height of ancestral tree.
`'height.dec'`	height of all descendent trees.
`'anc'`	an ancestral tree.
`'dec'`	all descendent trees.
`'org'`	an original tree.
`'equal'`	a three that descendants are all equal height.
`'max'`	a tree that descendants are scaled by the maximum height.
`'star'`	a tree that descendants are star trees.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`gen.equal.star.anc.dec`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# For gen.unit.K()
set.seed(1234)
tree.K <- gen.unit.K(6, rep(3:5, 2),
                     rate.anc = 0.7, rate.dec = 1.1)
X.class <- as.numeric(gsub("d(.)(.*)", "\\1",
                           tree.K$org$tip.label))

# Plot results
par(mfrow = c(2, 2))
plotnj(tree.K$org, X.class, type = "p",
       edge.width.class = 2, main = "org")
axis(1)
plotnj(tree.K$equal, X.class, type = "p",
       edge.width.class = 2, main = "equal")
axis(1)
plotnj(tree.K$max, X.class, type = "p",
       edge.width.class = 2, main = "max")
axis(1)
plotnj(tree.K$star, X.class, type = "p",
       edge.width.class = 2, main = "star")
axis(1)

## End(Not run)
```

`get.rooted.tree.height`

Get a Rooted Tree Height

Description

This function gets a rooted tree height, and only meaningful for a **ultrametric** tree which has the equal height from the root to all leaves.

Usage

```
get.rooted.tree.height(rooted.tree,
    tol = .Machine$double.eps^0.5)
```

Arguments

`rooted.tree`	a rooted tree in `Class phylo`.
`tol`	for `is.ultrametric` of **ape**.

Details

The `rooted.tree` should be in a `phylo` class of **ape**, and should be ultrametric that may be created by `ms`.

Value

Return the rooted tree height.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`ms`, `read.tree`, `as.phylo`, `is.ultrametric`, `rescale.rooted.tree`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
ret.ms <- ms(5, 1, opts = paste("-T", sep = " "))
tree.ms <- read.tree(text = ret.ms[3])
is.ultrametric(tree.ms)
get.rooted.tree.height(tree.ms)

## End(Not run)
```

`getcut.fun` *Tzeng's Method: Finding the Best Number of Clusters*

Description

For SNP sequences only, Tzeng's method (2005) uses an evolution approach to group haplotypes based on a deterministic transformation of haplotype frequency. This function find the best number of clusters based on Shannon information content.

Usage

```
getcut.fun(pp.org, nn, plot = 0)
```

Arguments

pp.org	frequency of haplotypes, sorted in decreasing order.
nn	number of haplotypes.
plot	illustrated in a plot.

Details

pp.org is summarized from X in haplo.post.prob, nn is equal to the number of rows of X.

This function is called by haplo.post.prob to determine the best guess of number of clusters. See Tzeng (2005) and Shannon (1948) for details.

Value

Return the best guess of number of clusters.

Author(s)

Jung-Ying Tzeng.

Maintain: Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Tzeng, J.Y. (2005) "Evolutionary-Based Grouping of Haplotypes in Association Analysis", *Genetics Epidemiology*, **28**, 220-231. http://www4.stat.ncsu.edu/~jytzeng/software.php

Shannon, C.E. (1948) "A mathematical theory of communication", *Bell System Tech J*, **27**, 379-423, 623-656.

See Also

haplo.post.prob.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

data.path <- paste(.libPaths()[1], "/phyclust/data/crohn.phy", sep = "")
my.snp <- read.phylip(data.path, code.type = "SNP")
ret <- haplo.post.prob(my.snp$org, ploidy = 1)
getcut.fun(sort(ret$haplo$hap.prob, decreasing = TRUE),
           nn = my.snp$nseq, plot = 1)

## End(Not run)
```

`haplo.post.prob` *Tzeng's Method: Haplotype Grouping for SNP Sequences*

Description

For SNP sequences only, Tzeng's method (2005) uses an evolution approach to group haplotypes based on a deterministic transformation of haplotype frequency. This is a modified version of original function, `haplo.score.RD.unphased.fun`.

Usage

```
haplo.post.prob(X, ploidy = 2, skip.haplo = 1e-07, K = NULL)
```

Arguments

`X`	sid matrix with N rows/sequences and L columns/sites.
`ploidy`	ploidy, no effect for phase known, keep consistence only.
`skip.haplo`	lower bound of haplotypes frequencies.
`K`	number of clusters.

Details

`X` should be a phase known haplotype data. For phase unknown and Tzeng's method (2006) are not tested yet.

If `K` is NULL, the result of `getcut.fun` will be used.

Value

See the original paper and source codes' documents for details. The function returns a list contains:

`'haplo'`	summarized data set in a list contains:

'haplotype'	unique haplotypes, dim = $N_{unique} \times L$.
'hap.prob'	frequency of haplotypes.
'post'	posterior probabilities of phase unknown haplotypes.
'hap1code'	unique ids of 'haplotype'.
'hap2code'	unique ids of 'haplotype', no effect if ploidy = 2.
'indx.subj'	id of subjects.

`'FD.id'`	unique ids of 'haplotype' for full dimension analysis.
`'RD.id'`	unique ids of 'haplotype' for reduced dimension analysis.
`'FD.post'`	posterior probabilities for full dimension analysis.
`'RD.post'`	posterior probabilities for reduced dimension analysis.
`'g.truncate'`	number of clusters

ToDo(s)

- test codes for phased unknown cases.

Author(s)

Jung-Ying Tzeng.

Maintain: Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Tzeng, J.Y. (2005) "Evolutionary-Based Grouping of Haplotypes in Association Analysis", *Genetics Epidemiology*, **28**, 220-231. http://www4.stat.ncsu.edu/~jytzeng/software.php

See Also

getcut.fun.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

data.path <- paste(.libPaths()[1], "/phyclust/data/crohn.phy", sep = "")
my.snp <- read.phylip(data.path, code.type = "SNP")
ret <- haplo.post.prob(my.snp$org, ploidy = 1)
str(ret)

## End(Not run)
```

---

ms *Generating Samples under a Wright-Fisher Neutral Model of Genetic Variation*

Description

This function modifies the original standalone code of `ms()` developed by Hudson (2002) for generating samples/coalescent trees under a Wright-Fisher neutral model.

Usage

```
ms(nsam = NULL, nreps = 1, opts = NULL, temp.file = NULL,
   tbs.matrix = NULL)
```

Arguments

nsam	number of samples/coalescent trees, usually greater than 2.
nreps	number of replications.
opts	options as the standalone version.
temp.file	temporary file for ms output.
tbs.matrix	a matrix for 'tbs' options given in opts.

Details

This function directly reuses the C code of ms by arguments as input from the opts. The options opts is followed from the original ms except nsam and nreps. Note that stdin, stdout, and pipe are all disable from opts.

For examples, options commonly used in **phyclust** are:

- "-T": generate trees in a neutral model.
- "-G": generate trees with a population growth rate, e.g. "-G 0.5".

These will return trees in a NEWICK format which can be read by the read.tree() of **ape** and passed to seqgen() to generate sequences.

temp.file allows users to specify ms output file themselves, but this file will not be deleted nor converted into R after the call to ms(). Users should take care the readings. By default, ms() uses a system temp file to store the output which is converted into R after the call and is deleted after converting.

tbs.matrix is a matrix to specify the values of tbs given in opts. See demo('simu_ms_tbs') for an example how to use this additional option. This option has been slightly tweaked by utilizing tbs options in the standalone ms. However, the output format is not the same as that in the standalone ms. Post-process is required with caution.

Value

This function returns a vector, and each element stores one line of STDOUT of ms() separated by newline. The vector stores in a class ms. The details of output format can found on the website http://home.uchicago.edu/~rhudson1/source.html and its manual.

Warning(s)

Carefully read the ms's original document before using the ms() function.

Author(s)

Hudson, R.R. (2002).

Maintain: Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Hudson, R.R. (2002) "Generating Samples under a Wright-Fisher Neutral Model of Genetic Variation", *Bioinformatics*, **18**, 337-338. http://home.uchicago.edu/~rhudson1/source.html

See Also

print.ms(), read.tree(), bind.tree(), seqgen().

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

ms()

# an ancestral tree
set.seed(1234)
(ret.ms <- ms(nsam = 3, opts = "-T -G 0.1"))
(tree.anc <- read.tree(text = ret.ms[3]))
tree.anc$tip.label <- paste("a", 1:K, sep = "")

# adjacent descendant trees to the ancestral tree
K <- 3
N <- 12
N.k <- c(3, 4, 5)
ms.dec <- NULL         # a list to store trees of ms
tree.dec <- NULL       # a list to store the trees in phylo class
tree.joint <- tree.anc
for(k in 1:K){
  ms.dec[[k]] <- ms(N.k[k], opts = "-T -G 1.0")
  tree.dec[[k]] <- read.tree(text = ms.dec[[k]][3])
  tree.dec[[k]]$tip.label <- paste("d", k, ".", 1:N.k[k], sep = "")
  tree.joint <- bind.tree(tree.joint, tree.dec[[k]],
                          where = which(tree.joint$tip.label ==
                                        paste("a", k, sep = "")))
}
str(tree.joint)

# plot trees
par(mfrow = c(2, 3))
plot(tree.anc, main = paste("anc (", K, ")", sep = ""))
axis(1)
for(k in 1:K){
  plot(tree.dec[[k]], main = paste("dec", k, " (", N.k[k], ")", sep = ""))
  axis(1)
}
plot(tree.joint, main = paste("joint (", N, ")", sep = ""))
axis(1)

# use tbs option (an example from msdoc.pdf by Hudson, R.R.)
tbs.matrix <- matrix(c(3.0, 3.5, 5.0, 8.5), nrow = 2)
ret <- ms(nsam = 5, nreps = 2, opts = "-t tbs -r tbs 1000",
          tbs.matrix = tbs.matrix)
print(ret)

## End(Not run)
```

`nid.aid.cid` *Transfer nids (0, 1, ..., 4) , aids (0, 1, ..., 21) and cids (0, 1, ..., 64)*

Description

Transfer nids (0, 1, ..., 4) , aids (0, 1, ..., 21) and cids (0, 1, ..., 64).

Usage

```
### S3 methods for a list, vector or matrix (default).
nid2aid(nidseq, start = 1, end = NULL, drop.gap = FALSE, byrow = TRUE)
nid2cid(nidseq, start = 1, end = NULL, drop.gap = FALSE, byrow = TRUE)
cid2aid(cidseq)
aid2acode(aidseq, lower.case = FALSE)
acode2aid(acodeseq)
```

Arguments

`nidseq`	a numerical vector contains nucleotide ids, 0, 1, 2, 3, or 4.
`cidseq`	a numerical vector contains codon ids, 0, 1, ..., or 64.
`aidseq`	a numerical vector contains amino acid ids, 0, 1, ..., or 21.
`acodeseq`	a character vector contains amino acid codes.
`start`	the start site to translate.
`end`	the end site to translate.
`drop.gap`	ignore gaps if TRUE.
`byrow`	advanced option, default = TRUE.
`lower.case`	transfer in lower cases.

Details

These functions are based on the internal object `.nucleotide`, `.codon`, `.amino.acid`, and `.genetic.code` to transfer sequences.

Value

`nid2aid` and `cid2aid` returns a numerical vector containing amino acid ids, and `nid2cid` returns a numerical vector containing codon ids, `aid2acode` returns a character vector containing amino acid codes, and `acode2aid` returns a numerical vector containing amino acid ids.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.nucleotide, .amino.acid, .codon, .genetic.code, code2nid.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

### Test S3 methods by a vector
a.vector <- c("A", "C", "G", "-", "T", "A")
code2nid(a.vector)
nid2cid(code2nid(a.vector))
cid2aid(nid2cid(code2nid(a.vector)))
nid2aid(code2nid(a.vector))
aid2acode(nid2aid(code2nid(a.vector)))
acode2aid(aid2acode(nid2aid(code2nid(a.vector))))

### Test S3 methods by a matrix
a.matrix <- rbind(a.vector, a.vector, a.vector)
code2nid(a.matrix)
nid2cid(code2nid(a.matrix))
cid2aid(nid2cid(code2nid(a.matrix)))
nid2aid(code2nid(a.matrix))
aid2acode(nid2aid(code2nid(a.matrix)))
acode2aid(aid2acode(nid2aid(code2nid(a.matrix))))

### Test S3 methods by a list
a.list <- list(a.vector, a.vector)
code2nid(a.list)
nid2cid(code2nid(a.list))
cid2aid(nid2cid(code2nid(a.list)))
nid2aid(code2nid(a.list))
aid2acode(nid2aid(code2nid(a.list)))
acode2aid(aid2acode(nid2aid(code2nid(a.list))))

## End(Not run)
```

---

paml.baseml *Phylogenetic Analysis by Maximum Likelihood for Nucleotide Sequences*

Description

This function modifies the original standalone code of baseml in PAML developed by Yang (1997) for phylogenetic analysis by maximum likelihood. This function provides a way to generate an ancestral tree for given central sequences clustered by phyclust.

Usage

```
paml.baseml(X, seqname = NULL, opts = NULL, newick.trees = NULL)
paml.baseml.control(...)
paml.baseml.show.default()
```

Arguments

X	nid matrix with N rows/sequences and L columns/sites.
seqname	sequence names.
opts	options as the standalone version, provided by paml.baseml.contol.
newick.trees	a vector/list contains NEWICK trees for runmode = 2.
...	for other possible opts and values. See PAML manual for details.
show	show opts and values.

Details

The function paml.baseml directly reuses the C code of baseml of PAML, and the function paml.baseml.control is to generate controls for paml.baseml as the file baseml.ctl of PAML.

The seqname should be consistent with X, and the leaf nodes of newick.trees.

The options opts is followed from the original baseml.ctl except seqfile, treefile and outputfile will be omitted.

paml.baseml.control generates default opts, and paml.baseml.show.default displays annotations for the default opts.

Value

This function returns a list, and each element stores one line of outputs of baseml separated by newline. The list stores in a class baseml. All the output of baseml of PAML will be saved in several files, and these will be read in by scan. Further post processing can be done by parsing the returning vector. The details of output format can found on the website http://abacus.gene.ucl.ac.uk/software/paml.html and its manual.

Note that some functionalities of baseml of PAML are changed in paml.baseml due to the complexity of input and output. The changes include such as disable the option G and rename the file 2base.t to pairbase.t.

Typically, the list contains the original output of baseml including pairbase.t, mlb, rst, rst1, and rub if they are not empty. The best tree (unrooted by default) will be stored in best.tree parsed from mlb based on the highest log likelihood. All output to STDOUT are stored in stdout. No STDIN are allowed.

Note that the print function for the class baseml will only show the best.tree only. Use str or names to see the whole returns of the list.

Warning(s)

Carefully read the PAML's original document before using the `paml.baseml` function, and `paml.baseml` may not be ported well from `baseml` of PAML. Please double check with the standalone version.

`baseml` may not be a well designed program, and may run slowly. If it were stuck, temporary files would all store at a directory obtained by `tempfile("/paml.baseml.")`.

`baseml` has its own options and settings which may be different than **phyclust** and **ape**. For example, the following is from the PAML's document, "In PAML, a rooted tree has a bifurcation at the root, while an unrooted tree has a trifurcation or multifurcation at the root." i.e. `paml.baseml` uses a rooted result for an unrooted tree, as well as for a rooted tree.

`baseml` also needs a sequence file which is dumped from R (duplicated from memory) for `paml.baseml`, and this file can be very big if sequences are too long or number of sequences is too large. Also, `paml.baseml` may take long time to search the best tree if data are large or initial trees are not provided.

Author(s)

Yang, Z. (1997) and Yang, Z. (2007)

Maintain: Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Yang, Z. (1997) "PAML: a program package for phylogenetic analysis by maximum likelihood", *Computer Applications in BioSciences*, **13**, 555-556.

Yang, Z. (2007) "PAML 4: a program package for phylogenetic analysis by maximum likelihood", *Molecular Biology and Evolution*, **24**, 1586-1591. http://abacus.gene.ucl.ac.uk/software/paml.html

See Also

`print.baseml`, `write.paml`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

paml.baseml.show.default()

### Generate data.
set.seed(123)
ret.ms <- ms(nsam = 5, nreps = 1, opts = "-T")
ret.seqgen <- seqgen(opts = "-mHKY -l40 -s0.2", newick.tree = ret.ms[3])
(ret.nucleotide <- read.seqgen(ret.seqgen))
X <- ret.nucleotide$org
seqname <- ret.nucleotide$seqname

### Run baseml.
```

```
opts <- paml.baseml.control(model = 4, clock = 1)
(ret.baseml <- paml.baseml(X, seqname = seqname, opts = opts))
(ret.baseml.init <- paml.baseml(X, seqname = seqname, opts = opts,
   newick.trees = ret.ms[3]))
ret.ms[3]

### Unrooted tree.
opts <- paml.baseml.control(model = 4)
(ret.baseml.unrooted <- paml.baseml(X, seqname = seqname, opts = opts))

### More information.
opts <- paml.baseml.control(noisy = 3, verbose = 1, model = 4, clock = 1)
ret.more <- paml.baseml(X, seqname = seqname, opts = opts)
# ret.more$stdout

### Plot trees
par(mfrow = c(2, 2))
plot(read.tree(text = ret.ms[3]), main = "true")
plot(read.tree(text = ret.baseml$best.tree), main = "baseml")
plot(read.tree(text = ret.baseml.init$best.tree), main = "baseml with initial")
plot(unroot(read.tree(text = ret.baseml.unrooted$best.tree)),
     main = "baseml unrooted")

## End(Not run)
```

---

phyclust *The Main Function of phyclust*

Description

The main function of **phyclust** implements finite mixture models for sequence data that the mutation processes are modeled by evolution processes based on Continuous Time Markov Chain theory.

Usage

```
phyclust(X, K, EMC = .EMC, manual.id = NULL, label = NULL, byrow = TRUE)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
K	number of clusters.
EMC	EM control.
manual.id	manually input class ids.
label	label of sequences for semi-supervised clustering
byrow	advanced option for X, default = TRUE.

Details

`X` should be a numerical matrix containing sequence data that can be transfered by `code2nid` or `code2sid`.

`EMC` contains all options used for EM algorithms.

`manual.id` manually input class ids as an initialization only for the initialization method, 'manualMu'.

`label` indicates the known clusters for labeled sequences which is a vector with length `N` and has values from `0` to `K`. `0` indicates clusters are unknown. `label` = `NULL` is for unsupervised clustering. Only un- and semi-supervised clustering are implemented.

`byrow` used in bootstraps to avoid transposing matrix 'X'. If FALSE, then the 'X' should be have the dimension $L \times K$.

Value

A list with class `phyclust` will be returned containing several elements as the following:

`'N.X.org'`	number of sequences in the `X` matrix.
`'N.X.unique'`	number of unique sequences in the `X` matrix.
`'L'`	number of sites, length of sequences, number of column of the `X` matrix.
`'K'`	number of clusters.
`'Eta'`	proportion of subpopulations, η_k, length = `K`, sum to 1.
`'Z.normalized'`	posterior probabilities, Z_{nk}, each row sums to 1.
`'Mu'`	centers of subpopulations, dim = $K \times L$, each row is a center.
`'QA'`	Q matrix array, information for the evolution model, a list contains:

'pi'	equilibrium probabilities, each row sums to 1.
'kappa'	transition and transversion bias.
'Tt'	total evolution time, t.
'identifier'	identifier for QA.

`'logL'`	log likelihood values.
`'p'`	number of parameters.
`'bic'`	BIC, $-2 \log L + p \log N$.
`'aic'`	AIC, $-2 \log L + 2p$.
`'N.seq.site'`	number of segregating sites.
`'class.id'`	class id for each sequences based on the maximum posterior.
`'n.class'`	number of sequences in each cluster.
`'conv'`	convergence information, a list contains:

'eps'	relative error.
'error'	error if the likelihood decreased.
'flag'	convergence state.

'iter'	convergence iterations.
'inner.iter'	convergence of inner iterations other than EM.
'cm.iter'	convergence of CM iterations.
'check.param'	parameter states.

`'init.procedure'`
: initialization procedure.

`'init.method'`
: initialization method.

`'substitution.model'`
: substitution model.

`'edist.model'`
: evolution distance model.

`'code.type'`
: code type.

`'em.method'`
: EM algorithm.

`'boundary.method'`
: boundary method.

`'label.method'`
: label method.

ToDo(s)

- make a general class for `Q` and `QA`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.EMC`, `.EMControl`, `find.best`, `phyclust.se`. `phyclust.se.update`.

Examples

```
library(phyclust, quiet = TRUE)

X <- seq.data.toy$org

set.seed(1234)
(ret.1 <- phyclust(X, 3))

EMC.2 <- .EMC
EMC.2$substitution.model <- "HKY85"
# the same as EMC.2 <- .EMControl(substitution.model = "HKY85")

(ret.2 <- phyclust(X, 3, EMC = EMC.2))
```

```
# for semi-supervised clustering
semi.label <- rep(0, nrow(X))
semi.label[1:3] <- 1
(ret.3 <- phyclust(X, 3, EMC = EMC.2, label = semi.label))
```

---

phyclust.e.step *One E-Step of phyclust*

Description

This is a single E-step of phyclust, usually following or followed by the other M-step.

Usage

```
phyclust.e.step(X, ret.phyclust = NULL, K = NULL, Eta = NULL,
    Mu = NULL, pi = NULL, kappa = NULL, Tt = NULL,
    substitution.model = NULL, identifier = NULL, code.type = NULL,
    Z.state = TRUE, label = NULL)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
ret.phyclust	an object with the class phyclust.
K	number of clusters.
Eta	proportion of subpopulations, η_k, length = K, sum to 1.
Mu	centers of subpopulations, dim = $K \times L$, each row is a center.
pi	equilibrium probabilities, each row sums to 1.
kappa	transition and transversion bias.
Tt	total evolution time, t.
substitution.model	substitution model.
identifier	identifier.
code.type	code type.
Z.state	see 'Details'.
label	label of sequences for semi-supervised clustering.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Either input ret.phyclust or all other arguments for this function except Z.state. ret.phyclust can be obtain either from an EM iteration of phyclust or from a M step of phyclust.m.step.

Z.state indicates the return values of Z_{nk}. If TRUE, the Z.normalized returned by this function will be posterior probabilities. Otherwise, it will be logPt, log of transition probabilities, $\log(\phi(\cdots))$.

If label is inputted, the label information will be used the E-step, even the ret.phyclust is the result of unsupervised clustering.

Value

This function returns a Z_{nk} matrix with dimension = $N \times K$. The values is dependent on Z.state, and they are either posterior probabilities if TRUE or transition probabilities otherwise.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, phyclust.em.step, phyclust.m.step.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)
X <- seq.data.toy$org

ret.1 <- phyclust(X, 2, EMC = EMC.1)
ret.2 <- phyclust.e.step(X, ret.phyclust = ret.1)
str(ret.2)

# For semi-supervised clustering.
semi.label <- rep(0, nrow(X))
semi.label[1:3] <- 1
ret.3 <- phyclust.e.step(X, ret.phyclust = ret.1, label = semi.label)

## End(Not run)
```

phyclust.edist *Evolution Distance of Sequences*

Description

This computes pair wised evolution distance of sequences.

Usage

```
phyclust.edist(X, edist.model = .edist.model[1])
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
edist.model	evolution distance model.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Value

This function returns a object with class dist.

ToDo(s)

- incorporate dist.dna of **ape**.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.edist.model.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

X <- rbind(c(0, 2, 1, 3, 0, 2, 2, 0, 3, 2, 2),
           c(0, 0, 1, 3, 2, 2, 1, 0, 3, 1, 2),
           c(0, 2, 1, 1, 0, 2, 1, 3, 0, 0, 1),
           c(2, 2, 1, 1, 0, 0, 2, 3, 0, 2, 1),
           c(2, 2, 1, 1, 0, 0, 2, 3, 1, 2, 0))
(ret <- phyclust.edist(X, edist.model = "D_HAMMING"))
str(ret)
as.matrix(ret)
plot(nj(ret), type = "u", no.margin = TRUE)

## End(Not run)
```

---

phyclust.em.step *One EM-step of phyclust*

Description

This is a single EM-step of phyclust.

Usage

```
phyclust.em.step(X, ret.phyclust = NULL, K = NULL, Eta = NULL,
    Mu = NULL, pi = NULL, kappa = NULL, Tt = NULL,
    substitution.model = NULL, identifier = NULL, code.type = NULL,
    label = NULL)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
ret.phyclust	an object with the class phyclust.
K	number of clusters.
Eta	proportion of subpopulations, η_k, length = K, sum to 1.
Mu	centers of subpopulations, dim = $K \times L$, each row is a center.
pi	equilibrium probabilities, each row sums to 1.
kappa	transition and transversion bias.
Tt	total evolution time, t.
substitution.model	substitution model.
identifier	identifier.
code.type	code type.
label	label of sequences for semi-supervised clustering.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Either input ret.phyclust or all other arguments for this function. ret.phyclust can be obtained either from an EM iteration of phyclust or from a M step of phyclust.m.step.

If label is inputted, the label information will be used the EM-step, even the ret.phyclust is the result of unsupervised clustering.

Value

This function returns an object with class phyclust.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, phyclust.e.step, phyclust.m.step.

Examples

```
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)
X <- seq.data.toy$org

ret.1 <- phyclust(X, 2, EMC = EMC.1)
ret.2 <- phyclust.em.step(X, ret.phyclust = ret.1)
str(ret.2)

# For semi-supervised clustering.
semi.label <- rep(0, nrow(X))
semi.label[1:3] <- 1
ret.3 <- phyclust.em.step(X, ret.phyclust = ret.1, label = semi.label)
str(ret.3)
```

phyclust.logL *Log-Likelihood of phyclust*

Description

This computes a log-likelihood value of phyclust.

Usage

```
phyclust.logL(X, ret.phyclust = NULL, K = NULL, Eta = NULL,
    Mu = NULL, pi = NULL, kappa = NULL, Tt = NULL,
    substitution.model = NULL, identifier = NULL, code.type = NULL,
    label = NULL)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
ret.phyclust	an object with the class phyclust.
K	number of clusters.
Eta	proportion of subpopulations, η_k, length = K, sum to 1.
Mu	centers of subpopulations, dim = $K \times L$, each row is a center.
pi	equilibrium probabilities, each row sums to 1.
kappa	transition and transversion bias.
Tt	total evolution time, t.
substitution.model	substitution model.
identifier	identifier.
code.type	code type.
label	label of sequences for semi-supervised clustering.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Either input ret.phyclust or all other arguments for this function. ret.phyclust can be obtain either from an EM iteration of phyclust or from a M step of phyclust.m.step.

If label is inputted, the label information will be used to calculate log likelihood (complete-data), even the ret.phyclust is the result of unsupervised clustering.

Value

This function returns a log-likelihood value of phyclust.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, phyclust.em.step.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)
X <- seq.data.toy$org

ret.1 <- phyclust(X, 2, EMC = EMC.1)
phyclust.logL(X, ret.phyclust = ret.1)

# For semi-supervised clustering.
semi.label <- rep(0, nrow(X))
semi.label[1:3] <- 1
phyclust.logL(X, ret.phyclust = ret.1, label = semi.label)

## End(Not run)
```

---

phyclust.m.step *One M-Step of phyclust*

Description

This is a single M-step of phyclust, usually following or followed by the other E-step.

Usage

```
phyclust.m.step(X, ret.phyclust = NULL, K = NULL,
    pi = NULL, kappa = NULL, Tt = NULL, Z.normalized = NULL,
    substitution.model = NULL, identifier = NULL, code.type = NULL,
    label = NULL)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
ret.phyclust	an object with the class phyclust.
K	number of clusters.
pi	equilibrium probabilities, each row sums to 1.
kappa	transition and transversion bias.
Tt	total evolution time, t.
Z.normalized	posterior probabilities obtained from an E-step.
substitution.model	substitution model.
identifier	identifier.
code.type	code type.
label	label of sequences for semi-supervised clustering.

Details

X should be a numerical matrix containing sequence data that can be transfered by code2nid or code2sid.

Either input ret.phyclust or all other arguments for this function. ret.phyclust can be obtained either from an EM iteration of phyclust or from a E step of phyclust.e.step.

If label is inputted, the label information will be used the M-step and Z.normalized will be replaced, even the ret.phyclust is the result of unsupervised clustering.

Value

This function returns an object with class phyclust.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, phyclust.em.step, phyclust.e.step.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$short.iter <- 1
EMC.1$EM.iter <- 1

# Test with phyclust.
X <- seq.data.toy$org
ret.1 <- phyclust(X, 2, EMC = EMC.1)

# Test with an em step.
ret.em <- phyclust.em.step(X, ret.1)

# Test with an E- and M-step.
ret.1$Z.normalized <- phyclust.e.step(X, ret.phyclust = ret.1)
ret.m <- phyclust.m.step(X, ret.phyclust = ret.1)

# Test with 2 em steps.
set.seed(1234)
EMC.2 <- EMC.1
EMC.2$EM.iter <- 2
ret.2 <- phyclust(X, 2, EMC = EMC.2)

# Check logL.
phyclust.logL(X, ret.1)
phyclust.logL(X, ret.em)
phyclust.logL(X, ret.m)
phyclust.logL(X, ret.2)

# For semi-supervised.
semi.label <- rep(0, nrow(X))
semi.label[1:3] <- 1
ret.m.1 <- phyclust.m.step(X, ret.phyclust = ret.1, label = semi.label)

## End(Not run)
```

`phyclust.Pt` *Transition Probabilities of phyclust Given Time*

Description

This computes transition probabilities of `phyclust` given time.

Usage

```
phyclust.Pt(Q, Tt, substitution.model = .substitution.model$model[1])
```

Arguments

`Q`	a list according to the substitution model.
`Tt`	total evolution time, t.
`substitution.model`	substitution model.

Details

The major models for `Q` are:

Model	Author and Publication	Parameter
JC69	Jukes and Cantor 1969.	t
K80	Kimura 1980.	κ, t
F81	Felsenstein 1981.	π, t
HKY85	Hasegawa, Kishino, and Yano 1985.	π, κ, t

A list of `Q` should contains `pi`, `kappa` based on substitution models and code types. `Tt` may be separately stored. Depending on identifiers, `Q`s can be composite to a `QA`, Q matrix array.

Value

A list with class `Pt` will be returned containing several elements as the following:

`'Pt'`	a transition probability matrix.
`'log.Pt'`	a log transition probability matrix.
`'H'`	a negative entropy, `diag(Pt %*% t(log.Pt))`.

ToDo(s)

- vectorize `Tt` for repeated computation in C.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.substitution.model`, `phyclust`, `phyclust.em.step`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

Tt <- 0.5

Q <- list(pi = c(0.25, 0.25, 0.25, 0.25), kappa = 0.5)
phyclust.Pt(Q, Tt, "HKY85")

Q <- list(pi = c(0.5, 0.5), kappa = 0.5)
phyclust.Pt(Q, Tt, "SNP_JC69")

## End(Not run)
```

---

phyclust.se *The Main Function of phyclust for Sequencing Error Models*

Description

The `phyclust.se` is an advanced function of `phyclust`. The `phyclust.se` implements finite mixture models for sequence data with sequencing errors. The same as `phyclust`, the mutation processes are modeled by evolution processes based on Continuous Time Markov Chain theory.

Usage

```
phyclust.se(X, K, EMC = .EMC, manual.id = NULL, byrow = TRUE)
```

Arguments

`X`	nid/sid matrix with N rows/sequences and L columns/sites.
`K`	number of clusters.
`EMC`	EM control.
`manual.id`	manually input class ids.
`byrow`	advanced option for `X`, default = TRUE.

Details

`phyclust.se` considers mutations with sequencing error, but only for NUCLEOTIDE data and only the EM algorithm is implemented. Currently, `phyclust.se` implements the following steps:

1. assume non-sequencing errors as the `phyclust` does.
2. use the initialization as the `phyclust` does.
3. run the `phyclust` to find the non-sequencing error MLE's.
4. initial by the results of `phyclust`.
5. update all parameters including probabilities of sequencing errors.

See the help page of phyclust for the explanations of input arguments since both functions share the same arguments. Note that the underling model assumptions are very different of both functions.

Value

A list with class phyclust will be returned containing several elements as described in phyclust. But, the phyclust.se returns extra parameters for sequencing errors, and they are shown in the following:

'SE' a list returning parameters of sequencing error models, including:

'type'	'TRUE' for modeling sequencing errors.
'model'	'CONVOLUTION', the only model implemented.
'constant'	the constrained constant for sequencing errors.
'f.err'	probability matrix, each row sums to 1.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.EMC, .EMControl, phyclust.se, phyclust.se.update.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

X <- seq.data.toy$org

set.seed(1234)
(ret.1 <- phyclust.se(X, 3))

## End(Not run)
```

---

phyclust.se.update *Update phyclust Results by the Sequencing Error Model*

Description

Since `phyclust.se` is difficult to optimize on a constrained high dimension parameter space, the `phyclust` is relatively easier to find a better result, as well as the `find.best` function.

This function will use the `phyclust` result as initial parameters and perform a sequencing error model. All parameters (Eta, Mu, Q, ...) in this function will be updated through the EM algorithm as `phyclust.se`.

Typically, this function run on the `find.best` results will yield a better result than on the `phyclust.se`.

Usage

```
phyclust.se.update(X, EMC = .EMC, ret.phyclust = NULL,
    K = NULL, Eta = NULL, Mu = NULL, pi = NULL, kappa = NULL,
    Tt = NULL, byrow = TRUE)
```

Arguments

`X`	nid/sid matrix with N rows/sequences and L columns/sites.
`EMC`	EM control.
`ret.phyclust`	an object with the class `phyclust`.
`K`	number of clusters.
`Eta`	proportion of subpopulations, η_k, length = `K`, sum to 1.
`Mu`	centers of subpopulations, dim = $K \times L$, each row is a center.
`pi`	equilibrium probabilities, each row sums to 1.
`kappa`	transition and transversion bias.
`Tt`	total evolution time, t.
`byrow`	advanced option for `X`, default = TRUE.

Details

All the input arguments are the same as the inputs of the function `phyclust.em.step` and `phyclust.update`.

Value

This function returns an object with class `phyclust`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`phyclust.se`, `phyclust.update`, `phyclust`, `find.best`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
X <- seq.data.toy$org

(ret.1 <- find.best(X, 4))
(ret.2 <- phyclust.se.update(X, ret.phyclust = ret.1))
.EMC$se.constant <- 1e-3
(ret.3 <- phyclust.se.update(X, ret.phyclust = ret.2))

### Search optimal error
func <- function(C){
  .EMC$se.constant <<- C
  -phyclust.se.update(X, ret.phyclust = ret.1)$logL
}
(ret.opt <- optimize(f = func, lower = 1e-3, upper = 1e-1))
.EMC$se.constant <- ret.opt$minimum
(ret.se.opt <- phyclust.se.update(X, ret.phyclust = ret.1))

## End(Not run)
```

`phyclust.update` *Update phyclust Results*

Description

This function will run the EM algorithm on initial parameters specified by users or from other initial procedures. All parameters (Eta, Mu, Q, ...) in this function will be updated.

Usage

```
phyclust.update(X, EMC = .EMC, ret.phyclust = NULL, K = NULL,
    Eta = NULL, Mu = NULL, pi = NULL, kappa = NULL, Tt = NULL,
    label = NULL, byrow = TRUE)
```

Arguments

`X`	nid/sid matrix with N rows/sequences and L columns/sites.
`EMC`	EM control.
`ret.phyclust`	an object with the class `phyclust`.
`K`	number of clusters.
`Eta`	proportion of subpopulations, η_k, length = `K`, sum to 1.
`Mu`	centers of subpopulations, dim = $K \times L$, each row is a center.
`pi`	equilibrium probabilities, each row sums to 1.

kappa	transition and transversion bias.
Tt	total evolution time, t.
label	label of sequences for semi-supervised clustering.
byrow	advanced option for X, default = TRUE.

Details

This function is equivalent to run exhaustEM on one specified initial parameters, and no initial procedure is involved. While this function is a little bit different to run phyclust with manual.id where Mu will be reestimated as the new initials. Simply speaking, this function only runs the EM algorithm given the initial parameters.

All the input arguments are the same as the inputs of the functions phyclust and phyclust.em.step.

Value

This function returns an object with class phyclust.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, find.best, phyclust.se, phyclust.se.update.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
EMC.1 <- .EMC
EMC.1$EM.iter <- 1
# the same as EMC.1 <- .EMControl(EM.iter = 1)
X <- seq.data.toy$org

(ret.1 <- phyclust(X, 2, EMC = EMC.1))
(ret.2 <- phyclust.update(X, ret.phyclust = ret.1))

## End(Not run)
```

plotdots *Dots Plots of Sequences for Visual Comparisons*

Description

This function provides dots plots of data set given an idea how diverse the sequences are by drawing dots with different colors for all mutations.

Usage

```
plotdots(X, X.class = NULL, Mu = NULL, code.type = .code.type[1],
    diff.only = TRUE, fill = FALSE, label = TRUE, with.gap = FALSE,
    xlim = NULL, ylim = NULL, main = "Dots Plot", xlab = "Sites",
    ylab = "Sequences", missing.col = "gray95", ...)
```

Arguments

X	numerical data matrix with N rows/sequences and L columns/sites.
X.class	class ids indicated for all sequences.
Mu	a center sequence with length L.
code.type	either "NUCLEOTIDE" (default) or "SNP".
diff.only	draw the segregating sites only, default = TRUE.
fill	fill in all dots, default = FALSE.
label	indicate segregating sites, default = TRUE.
with.gap	pass to find.consensus if Mu is NULL, default = FALSE
xlim	limit of x-axis.
ylim	limit of y-axis.
main	main label, default = "Dots Plot".
xlab	x-axis label, default = "Sites".
ylab	y-axis label, default = "Sequences".
missing.col	color for the missing allele, default = NA.
...	other options passed to plot.

Details

The first rows in Mu will be drawn entirely on dots plots in colors which are "green3", "blue2", "\#CC00CC", "red2", "gray", and "white", according the ids + 1. If fill is FALSE, other sequences will be drawn by the mutation sites comparing to the first sequences. Otherwise, they be drawn entirely.

If X.class is set, the sequences will be drawn in cluster order.

If Mu is NULL, the consensus sequence of X will be drawn.

If label is TRUE, the bottom row will be drawn in color "orange" to indicate segregating sites.

with.gap is only used when Mu is NULL.

Value

A dots plot will be drawn.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

seqgen, plothist.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# For nucleotide
X <- seq.data.toy$org
par(mfrow = c(2, 2))
plotdots(X)
plotdots(X, diff.only = FALSE)
plotdots(X, diff.only = FALSE, label = FALSE)
plotdots(X, fill = TRUE, diff.only = FALSE, label = FALSE)

# With class ids
X.class <- as.numeric(gsub(".*-(.*)", "\\1", seq.data.toy$seqname))
plotdots(X, X.class)

# For SNP
X.SNP <- nid2sid(X)
plotdots(X.SNP, X.class)

## End(Not run)
```

---

plotgaps *Gaps Plots of Sequences for Visual Comparisons*

Description

This function provides gaps plots of data set to identify regions where gaps enriched. The plot show the proportions of context by sites and the diverse may be caused by mutations, sequencing errors, or alignment errors.

Usage

```
plotgaps(X, code.type = .code.type[1], main = "Gaps Plot",
    xlab = "Sites", ylab = "Proportion", ...)
```

Arguments

X	numerical data matrix with N rows/sequences and L columns/sites.
code.type	either "NUCLEOTIDE" (default) or "SNP".
main	main label, default = "Gaps Plot".
xlab	x-axis label, default = "Sites".
ylab	y-axis label, default = "Proportion".
...	other options passed to plot.

Details

Proportions of gaps will be drawn.

Value

A gaps plot will be drawn.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

plotdots.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# For nucleotide
set.seed(1234)
X <- seq.data.toy$org
X[sample(c(T, F), length(X), replace = TRUE, prob = c(0.05, 0.95))] <-
  .nucleotide$nid[.nucleotide$code == "-"]
plotgaps(X)

## End(Not run)
```

plothist *Plot Histogram to Compare Number of Mutations.*

Description

Plot histogram to compare number of mutations.

Usage

```
plothist(X, X.class = NULL, Mu = NULL, fill.color = .Color,
   draw.all = TRUE, main = "Mutation counts",
   xlab = "Difference", ylab = "Counts", append = FALSE)
```

Arguments

X	nid/sid matrix with N rows/sequences and L columns/sites.
X.class	class ids indicated for all sequences.
Mu	a central sequence with length L.
fill.color	color to fill the histogram.
draw.all	draw a histogram use all sequences.
main	main label, default = "Mutation counts".
xlab	x-axis label, default = "Difference".
ylab	y-axis label, default = "Counts".
append	overwrite histograms.

Details

If X.class is set, the histograms will be drawn by classes and all sequences will be compared to the central sequence Mu. Otherwise, all sequences will be used to count mutations. draw.all is not effect if X.class is not set.

If Mu is set, it will be used to compare to all other sequences to count mutations. Otherwise, the first sequence of X will be used, and the first sequence in the first class will be used if X.class is set. If Mu is a matrix, the first row will be used as the central sequence.

Value

Histograms will be drawn to show the number of mutations away from the central sequence.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

seqgen, plotdots.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

X <- seq.data.toy$org
plothist(X)

# With class ids
X.class <- as.numeric(gsub(".*-(.*)", "\\1", seq.data.toy$seqname))
plothist(X, X.class)

## End(Not run)
```

---

plotnj *Plot an Unrooted Trees.*

Description

This is a enhanced version of `plot.phylo` in **ape** which can plot trees in `Class phylo` including neighbor-joining trees, unrooted trees, trees with star shapes, ... etc.

Usage

```
plotnj(unrooted.tree, X.class = NULL, type = "u", main = NULL,
   show.tip.label = FALSE, show.node.label = FALSE,
   edge.width = 1, edge.width.class = edge.width, ...)
```

Arguments

`unrooted.tree`	an unrooted tree in `Class phylo`.
`X.class`	class ids indicated for all tips.
`type`	plot types, see `plot.phylo` in **ape** for details.
`main`	main label.
`show.tip.label`	show tip label if available.
`show.node.label`	show node label if available.
`edge.width`	edge width for all internal branches if `X.class` is set.
`edge.width.class`	edge width for tip branches if `X.class` is set.
`...`	other options passed to `plot.phylo`.

Details

This function is built to plot unrooted trees, but it may also apply for other trees in `Class phylo`.

`type` can be "u", "p", "c", "f", "r" as in `plot.phylo`.

If `X.class` is set, then the tip branches will be drawn with colors by class ids, and the colors are controlled by `.color`. The width of branches is controlled by `edge.width` for all internal branches and by `edge.width.class` for tip branches.

Value

Return a tree plot.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`plot.phylo`, `.Color`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
ret.ms <- ms(nsam = 24, opts = "-T -G 0.5")
tree.anc <- read.tree(text = ret.ms[3])

is.rooted(tree.anc)
tree.new <- as.star.tree(tree.anc)
X.class <- rep(1:6, each = 4)

par(mfrow = c(2, 2))
plotnj(tree.anc, X.class, type = "u", edge.width.class = 2,
       main = "unrooted tree")
plotnj(tree.new, X.class, type = "u", edge.width.class = 2,
       main = "star tree")
plotnj(tree.anc, X.class, type = "c", edge.width.class = 2,
       main = "unrooted tree in cladogram")
plotnj(tree.new, X.class, type = "r", edge.width.class = 2,
       main = "star tree in radial")

## End(Not run)
```

plotstruct *Struct Plots of Observations Based on Posterior Probabilities*

Description

This function provides structure plots of data set given based on posterior probabilities, the `Z.normalized` matrix.

Usage

```
plotstruct(Z, X.class = NULL, sort.inside.class = TRUE,
    direction = "h", main = "Structure Plot", xlab = "Observations",
    ylab = "Posterior Probabilities", ...)
```

Arguments

`Z`	a Z matrix as Z.normalized in `Class phyclust`.
`X.class`	class ids indicated for all observations
`sort.inside.class`	sort observations inside class by max posteriors.
`direction`	either "h" for horizontal or "v" for vertical.
`main`	main label, default = "Structure Plot".
`xlab`	x-axis label, default = "Observations".
`ylab`	y-axis label, default = "Posterior Probabilities".
`...`	other options passed to `plot`.

Details

The posterior probabilities in `ret.phyclust$Z.normalized` will be drawn in colors.

If `X.class` is submitted, the plot will draw in the order of class ids and the `sort.inside.class` will be skipped.

Value

A structure plot will be drawn.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

phyclust, find.best, plotdots.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
ret.1 <- phyclust(seq.data.toy$org, 3)
plotstruct(ret.1$Z.normalized)
windows()
plotstruct(ret.1$Z.normalized, sort.inside.class = FALSE)

# With class ids
X.class <- as.numeric(gsub(".*-(.*)", "\\1", seq.data.toy$seqname))
windows()
plotstruct(ret.1$Z.normalized, X.class = X.class)

## End(Not run)
```

---

print.object *Functions for Printing or Summarizing Objects According to Classes*

Description

Several classes are declared in **phyclust**, and these are functions to print and summary objects.

Usage

```
## S3 method for class 'baseml'
print(x, ...)
## S3 method for class 'ms'
print(x, ...)
## S3 method for class 'phyclust'
print(x, digits = max(4, getOption("digits") - 3), ...)
## S3 method for class 'Pt'
print(x, ...)
## S3 method for class 'RRand'
print(x, digits = max(4, getOption("digits") - 3), ...)
## S3 method for class 'seq.data'
print(x, ...)
## S3 method for class 'seqgen'
print(x, ...)
## S3 method for class 'phyclust'
summary(object, ...)
```

Arguments

x	an object with the class attributes.
digits	for printing out numbers.
object	an object with the class attributes.
...	other possible options.

Details

These are useful functions for summarizing and debugging.

For ms, seqgen, and paml.baseml, it will show the result as standalone versions on the STDOUT out with line by line.

For other functions, they only show summaries of objects. Use names or str to explore the details.

Value

The results will cat or print on the STDOUT by default.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

ms, paml.baseml, phyclust, phyclust.Pt, RRand, seqgen.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

# Functions applied by directly type the names of objects.

## End(Not run)
```

`prune.Mu` *Prune the Center Sequences Mu*

Description

This function prune the center sequences Mu where the sites will be reset as GAPs if all members within the same cluster are all GAPs.

Usage

```
prune.Mu(X, X.class, Mu, code.type = .code.type[1])
```

Arguments

`X`	numerical data matrix with N rows/sequences and L columns/sites.
`X.class`	class ids indicated for all sequences.
`Mu`	a center sequence with length L.
`code.type`	either "NUCLEOTIDE" (default) or "SNP".

Details

For each cluster indicated by `X.class`, this function will prune `Mu` and reset the sites as GAPs if all members within cluster are all GAPs. `Mu` are usually the returning values of `phyclust()`.

Value

A pruned `Mu` will be returned.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`phyclust`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

X <- seq.data.toy$org
X[, 5] <- .nucleotide$nid[.nucleotide$code == "-"]
ret <- phyclust(X, 2)
Mu.GAPs <- prune.Mu(X, ret$class.id, ret$Mu)

ret$Mu[, 5]
Mu.GAPs[, 5]  # Replace by GAPs.

## End(Not run)
```

`read.seqgen` *Read seqgen's Results and Return a seq.data*

Description

This function can read the results generated by `seqgen` and turn into a object in `Class seq.data`.

Usage

```
read.seqgen(text, byrow = TRUE, code.type = .code.type[1])
```

Arguments

`text`	a text vector generated by `seqgen`.
`byrow`	advanced option, default = TRUE.
`code.type`	either "NUCLEOTIDE" (default) or "SNP".

Details

If `code.type` is "SNP", the A, G will be transfered to 1, and the C, T will be transfered to 2.

Value

Return an object in `Class seq.data`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

seqgen, gen.seq.HKY, gen.seq.SNP.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(123)
ret.ms <- ms(nsam = 5, nreps = 1, opts = "-T")
ret.seqgen <- seqgen(opts = "-mHKY -l40 -s0.2", newick.tree = ret.ms[3])
(ret.nucleotide <- read.seqgen(ret.seqgen))
(ret.snp <- read.seqgen(ret.seqgen, code.type = "SNP"))

## End(Not run)
```

---

rescale.rooted.tree *Rescale a Rooted Tree's Height*

Description

This function rescaled the input rooted tree height by a scale.height.

Usage

```
rescale.rooted.tree(rooted.tree, scale.height = 1)
```

Arguments

rooted.tree	a rooted tree in Class phylo.
scale.height	a scale to all branch lengths.

Details

The rooted.tree should be in a phylo class of **ape**, and may be created by ms.

scale.height is a positive number multiplying on the lengths of all branches of the rooted tree.

Value

Return a rooted tree in Class phylo with scaled branches.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

ms, read.tree, as.phylo, get.rooted.tree.height.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(1234)
ret.ms <- ms(5, 1, opts = paste("-T", sep = " "))
tree.ms <- read.tree(text = ret.ms[3])
get.rooted.tree.height(tree.ms)

tree.scaled <- rescale.rooted.tree(tree.ms, scale.height = 2)
get.rooted.tree.height(tree.scaled)

## End(Not run)
```

RRand *Rand Index and Adjusted Rand Index*

Description

This function returns the Rand index and the adjusted Rand index for given true class ids and predicted class ids.

Usage

```
RRand(trcl, prcl, lab = NULL)
```

Arguments

trcl	true class ids.
prcl	predicted class ids.
lab	known ids for semi-supervised clustering.

Details

All ids, trcl and prcl, should be positive integers and started from 1 to K, and the maximums are allowed to be different.

lab used in semi-supervised clustering contains the labels which are known before clustering. It should be positive integer and started from 1 for labeled data and 0 for unlabeled data.

Value

Return a Class RRand contains Rand index and adjusted Rand index.

Author(s)

Ranjan Maitra.

Maintain: Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

true.id <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
pred.id <- c(2, 1, 2, 1, 1, 1, 2, 1, 1)
label   <- c(0, 0, 0, 0, 1, 0, 2, 0, 0)

RRand(true.id, pred.id)
RRand(true.id, pred.id, lab = label)

## End(Not run)
```

---

seq.data *A Toy Dataset in Class seq.data*

Description

A toy dataset, `seq.data.toy`, with 100 nucleotide sequences and 200 sites in 4 clusters. `seq.data.gap` contains some missing values indicated by "-".

Format

This data contains a list with a `seq.data` structure described in the 'Details' section.

Details

A toy dataset is generated to demonstrate **phyclust**. It has 100 nucleotide sequences and 200 sites in 4 clusters where the ancestral tree height 0.15 and the descendant tree height 0.09, and sequences are evolved by a HKY85 model.

The structre of class `seq.data` is a list containing:

`code.type`	either "NUCLEOTIDE" or "SNP".
`info`	header for phylip file.
`nseq`	number of sequences, N.
`seqlen`	length of sequences, L.
`seqname`	sequence names.
`org.code`	original codes, dim = $N \times L$.

org	transfered ids, dim = $N \times L$.
byrow	TRUE for dim = $N \times L$, FALSE for transpose.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

print.seq.data.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

seq.data.toy
seq.data.gap

par(mfrow = c(1, 2))
plotdots(seq.data.toy$org)
plotdots(seq.data.gap$org)

## End(Not run)
```

---

seqgen *Seq-Gen*

Description

This function modifies the original standalone code of seq-gen developed by Rambaut, A. and Grassly, N.C. (1997).

Usage

```
seqgen(opts = NULL, rooted.tree = NULL, newick.tree = NULL, input = NULL,
       temp.file = NULL)
```

Arguments

opts	options as the standalone version.
rooted.tree	a rooted tree which sequences are generated according to.
newick.tree	a NEWICK tree which sequences are generated according to.
input	optional inputs of seq-gen, e.g. ancestral sequences.
temp.file	temporary file for seqgen output.

Details

This function directly reuses the C code of `seq-gen` by arguments as input from the STDIN. The options `opts` is followed from the original `seq-gen` except an input tree.

Input either a `rooted.tree` or a `newick.tree`, and `rooted.tree` should have a `Class` `phylo`.

For examples, options commonly used in **phyclust** are:

- "-m": set an evolution model, e.g. "-mHKY".
- "-t": set transition/transversion ratio, e.g. "-t0.7".
- "-f": equilibrium probabilities of A, C, G, and T, e.g. "-f0.1,0.2,0.3,0.4".
- "-l": length of sequences, e.g. "-l10".
- "-s": scale rate for the total height of input tree, "-s0.2".
- "-k": index of ancestral sequence in input file, see `gen.seq.HKY`.

These will return sequences in `Format` `phylip` which can be read by `read.seqgen()` and transfered into an object with `Class` `seq.data`.

The maximum number of tips is 2000 in `seqgen()` by default, but an extra option `opts = "-u 2014 ..."` can be simply increase the size to 2014.

Note:

- `input` and `rooted.tree`/`newick.tree` can not be submitted at the same time.
- `seq-gen` use the order A, C, G, T.
- `-t` is ts/tv ratio which is not equal to κ.
- See more examples in `gen.seq.HKY()` and `gen.seq.SNP()`.

`temp.file` allows users to specify seqgen output file themselves, but this file will not be deleted nor converted into R after the call to `seqgen()`. Users should take care the readings. By default, `seqgen()` uses a system temp file to store the output which is converted into R after the call and is deleted after converting.

Value

This function returns a vector, and each element stores one line of STDOUT of `seq-gen` separated by newline. The vector stores in a class `seqgen`. The details of output format can found on the website http://tree.bio.ed.ac.uk/software/seqgen/ and its manual.

Warning(s)

Carefully read the `seq-gen`'s original document before using the `seqgen()` function.

Author(s)

Rambaut, A. and Grassly, N.C. (1997).

Maintain: Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

Rambaut, A. and Grassly, N.C. (1997) "Seq-Gen: An Application for the Monte Carlo Simulation of DNA Sequence Evolution along Phylogenetic Trees", *Computer Applications In The Biosciences*, **13:3**, 235-238. http://tree.bio.ed.ac.uk/software/seqgen/

See Also

print.seqgen(), read.tree(), ms(), gen.seq.HKY(), gen.seq.SNP().

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

set.seed(123)
ret.ms <- ms(nsam = 5, nreps = 1, opts = "-T")
seqgen(opts = "-mHKY -l40 -s0.2", newick.tree = ret.ms[3])


## End(Not run)
```

---

snp2sid *Transfer SNP codes (1, 2, -) and sids (0, 1, 2)*

Description

Transfer SNP codes (1, 2, -) and SNP ids (0, 1, 2).

Usage

```
snp2sid(snpseq)
sid2snp(sidseq)
```

Arguments

snpseq	a character vector contains SNP codes, 1, 2, or -.
sidseq	a numerical vector contains SNP ids, 0, 1, or 2.

Details

This function is based on the internal object .snp to transfer SNP codes and SNP ids.

Value

snp2sid returns a numerical vector containing SNP ids, and sid2snp returns a character vector containing SNP codes.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

.show.option, .snp, code2nid, nid2code, code2snp, snp2code.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

a <- c("1", "2", "1", "-", "2")
snp2sid(a)
sid2snp(snp2sid(a))

## End(Not run)
```

---

standard.code *Standard Codes and ids for Nucleotides, SNPs, Codon, Amino Acid and Genetic Code*

Description

Standard codes and ids for nucleotides, SNPs, codon, amino acid and genetic code. All objects are used to transfer data. **These are read-only objects and the elemental order is followed in C.**

Usage

```
.nucleotide
.snp
.codon
.amino.acid
.genetic.code
```

Format

All objects are data frames containing ids and codes.

Details

Note: All ids are coding started from **0**.

Nucleotides, A, G, C, T, and - have codes 0, 1, 2, 3, and 4 where "-" is for gaps. SNPs, 1, 2, and - have code codes 0, 1, and 2.

These are objects are in data frames unlike other internal objects due to heavily used in processing data. The original data should be transfered to numerical codes in order to be passed to C codes. In C codes, we use integers, 0, 1, 2, ..., for coding nucleotides or SNPs and so on.

Now, models and methods are implemented only for `.nucleotide` and `.snp`. Other objects are leaved for further extension.

Data frames use factor formats as the default, and `as.character` is the way to transfer to the characters.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com>

References

Phylogenetic Clustering Website: https://snoweye.github.io/phyclust/

See Also

`.show.option`, `code2nid`, `nid2code`, `snp2sid`, `sid2snp`.

Examples

```
## Not run:
library(phyclust, quiet = TRUE)

.nucleotide
.snp
.codon
.amino.acid
.genetic.code
.missing.code

## End(Not run)
```

Index