Package: poweRbal (via r-universe)

June 29, 2024

Title Phylogenetic Tree Models and the Power of Tree Shape Statistics

Version 0.0.0.3

Author Sophie Kersting [aut, cre]
(<https://orcid.org/0000-0002-1038-9246>), Kristina Wicke [aut]
(<https://orcid.org/0000-0002-4275-5546>), Mareike Fischer
[aut] (<https://orcid.org/0000-0002-9429-0859>)

Maintainer Sophie Kersting <sophie_kersting@gmx.de>

Description The first goal of this package is to provide a multitude
of tree models, i.e., functions that generate rooted binary
trees with a given number of leaves. Second, the package allows
for an easy evaluation and comparison of tree shape statistics
by estimating their power to differentiate between different
tree models. Please note that this R package was developed
alongside the manuscript ``Tree balance in phylogenetic models''
by S. J. Kersting, K. Wicke, and M. Fischer (2024)
<doi:10.48550/arXiv.2406.05185>, which provides further
background and the respective mathematical definitions. This
project was supported by the project ArtIGROW, which is a part
of the WIR!-Alliance ArtIFARM – Artificial Intelligence in
Farming funded by the German Federal Ministry of Education and
Research (No. 03WIR4805).

License GPL (>= 3)

Depends R (>= 3.5.0)

Imports ape, scales, phytools, treebalance, R.utils, diversitree

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.1

Repository CRAN

Date/Publication 2024-06-28 09:20:02 UTC

Contents

`enum2cladewise` *Function to modify the node enumeration in rooted binary trees*

Description

`enum2cladewise` - Changes the node enumeration to cladewise enumeration, i.e., starting from the root we follow the rule:
Go to the left child; if that does not exist or was already visited go (up again and) to the right child. The nodes in the rooted binary tree can be nearly arbitrarily enumerated (distinct nodes should have distinct values and the values should be positive, i.e., >0).

Usage

```
enum2cladewise(phy, root = NULL)
```

Arguments

`phy`	A rooted binary tree of class `phylo`.
`root`	Integer value (default = NULL) that should only be specified if the root is known precisely (not necessary, but speeds up computation).

Value

enum2cladewise A single tree of class phylo is returned with cladewise node enumeration.

Examples

```
# Example with cladewise enumeration:
phy_alreadycladew <- list(edge = matrix(c(6,7, 7,8, 8,1, 8,2,
                                          7,9, 9,3, 9,4, 6,5),
                                  byrow = TRUE, ncol = 2),
                          tip.label = rep(" ",5), Nnode = 4)
attr(phy_alreadycladew, "class") <- "phylo"
enum2cladewise(phy_alreadycladew, root = 6)$edge
ape::plot.phylo(phy_alreadycladew)
# Example with other node enumeration:
phy_example <- list(edge = matrix(c(1,55, 55,12, 12,2, 12,10, 55,9,
                                    9,13, 9,60, 1,3),
                                  byrow = TRUE, ncol = 2),
                    tip.label = rep(" ",5), Nnode = 4,
                    edge.length = rep(1, 8))
attr(phy_example, "class") <- "phylo"
# The reenumeration works with and without specifying the root:
enum2cladewise(phy_example, root = 1)$edge
ape::plot.phylo(enum2cladewise(phy_example))
```

---

genAldousBetaTree *Generation of rooted binary trees under Aldous' beta splitting model*

Description

genAldousBetaTree - Generates a rooted binary tree in phylo format with the given number of n leaves under the Aldous beta model. The Aldous beta model is not a rate-based incremental evolutionary (tree) construction and thus cannot generate edge lengths, only a topology. Instead, the Aldous beta model works as follows: The idea is to start with the root and the set of its descendant leaves, i.e., all n leaves. Then, this set is partitioned into two subsets according to a density function dependent on the parameter beta. The two resulting subsets contain the leaves of the two maximal pending subtrees of the root, respectively. The same procedure is then applied to the root's children and their respective subsets, and so forth.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genAldousBetaTree(n, BETA)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

BETA — Numeric value >=-2 which specifies how the leaf sets are partitioned. For certain choices of BETA the Aldous beta model coincides with known models:

- BETA = 0: Yule model
- BETA = -3/2: PDA model (all phylogenies equally probable)
- BETA = -2: Caterpillar with n leaves

Value

genAldousBetaTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- D. Aldous. Probability Distributions on Cladograms. In Random Discrete Structures, pages 1–18. Springer New York, 1996.

Examples

```
genAldousBetaTree(n = 5, BETA = 1)
```

---

genAltBirthDeathTree *Generation of rooted binary trees under the alternative birth-death model*

Description

genAltBirthDeathTree - Generates a rooted binary tree in phylo format with the given number of n leaves under the alternative birth-death model. In the alternative birth-death process all species have the same speciation BIRTHRATE and extinction rates DEATHRATE. Extinct species remain as fossils inside the tree with zero speciation and extinction rates.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genAltBirthDeathTree(n, BIRTHRATE = 1, DEATHRATE = 0, TRIES = 5)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
BIRTHRATE	Positive numeric value (default = 1) which specifies the rate at which the speciation events occur.
DEATHRATE	Positive numeric value (default = 0) which specifies the rate at which the extinction events occur.
TRIES	Integer value (default = 5) that specifies the number of attempts to generate a tree with n leaves.

Value

genAltBirthDeathTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models. arXiv:2406.05185, 2024.
- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models: Supplementary material. https://tinyurl.com/278cwdh8, 2024.

Examples

```
genAltBirthDeathTree(n = 7, DEATHRATE = 1)
```

---

genBiSSETree *Generation of rooted binary trees under the BiSSE model*

Description

genBiSSETree - Generates a rooted binary tree in phylo format with the given number of n leaves under the BiSSE model. In the BiSSE model all species have a state, either A or B, and depending on the state a speciation rate BIRTHRATES, an extinction rate DEATHRATES as well as a transition rate to the other state TRANSRATES.
Extinct species are removed from the tree, i.e., the generated tree contains only species living at the present.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genBiSSETree(
  n,
  BIRTHRATES = c(1, 1),
  DEATHRATES = c(0, 0),
  TRANSRATES,
  TRIES = 5,
  TIMEperTRY = 0.1
)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
BIRTHRATES	Numeric vector (default = c(1,1)) which specifies the speciation rates in state A and B (vector with 2 values >=0, one value >0).
DEATHRATES	Numeric vector (default = c(0,0)) which specifies the extinction rates in state A and B (vector with 2 values >=0).
TRANSRATES	Numeric vector which specifies the transition rates from A to B and from B to A (vector with 2 values >0).
TRIES	Integer value (default = 5) that specifies the number of attempts to generate a tree with n leaves.
TIMEperTRY	Numeric value (default = 0.1) that specifies the maximum amount of time (in seconds) invested per try.

Value

genBiSSETree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- This function uses the tree.bisse function of the diversitree package (R. G. FitzJohn. Diversitree: Comparative Phylogenetic Analyses of Diversification in R. Methods in Ecology and Evolution, 3(6):1084-1092, 2012).

- W. P. Maddison, P. E. Midford, and S. P. Otto. Estimating a binary character's effect on speciation and extinction. Systematic Biology, 56(5):701–710, 2007.

Examples

```
if (requireNamespace("diversitree", quietly = TRUE)) {
genBiSSETree(n = 5, BIRTHRATES = c(1,2), DEATHRATES = c(0,0),
             TRANSRATES = c(0.1,0.3))
}
```

genCombTree *Generation of the comb or caterpillar tree*

Description

genCombTree - Generates the rooted binary comb tree (also known as caterpillar tree) in phylo format with the given number of n leaves.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genCombTree(n)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

Value

genCombTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- D. Aldous. Probability Distributions on Cladograms. In Random Discrete Structures, pages 1–18. Springer New York, 1996.

Examples

```
genCombTree(n = 6)
```

genDensityTree *Generation of rooted binary trees under the density model*

Description

genDensityTree - Generates a rooted binary tree in phylo format with the given number of n leaves under the density-dependent model. In the density-dependent tree generation process all species have the same speciation BIRTHRATE, but the extinction rates depend on the number of species (it increases linearly with the number of co-existing lineages until an equilibrium number is reached at which speciation and extinction rates are equal). Extinct species are removed from the tree, i.e., the generated tree contains only species living at the present.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genDensityTree(n, BIRTHRATE = 1, EQUILIB, TRIES = 5, TIMEperTRY = 0.01)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
BIRTHRATE	Positive numeric value (default = 1) which specifies the rate at which the speciation events occur.
EQUILIB	Integer value that specifies the equilibrium number.
TRIES	Integer value (default = 5) that specifies the number of attempts to generate a tree with n leaves.
TIMEperTRY	Numeric value (default = 0.01) that specifies the maximum amount of time (in seconds) invested per try.

Value

genDensityTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- P. H. Harvey, R. M. May, and S. Nee. Phylogenies without fossils. Evolution, 48(3):523–529, 1994.

Examples

```
genDensityTree(n = 5, EQUILIB = 6)
```

genETMTree *Generation of rooted binary trees under the equiprobable-types-model (ETM)*

Description

genETMTree - Generates a rooted binary tree in phylo format with the given number of n leaves under the equiprobable-types-model. Given n, all tree shapes/topologies with n leaves are equiprobable under the ETM.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genETMTree(n)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

Value

genETMTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- This function uses the rtree(..., equiprob = T) function of the ape package (E. Paradis, K. Schliep. "ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R." Bioinformatics, 35, 526-528, 2019).

Examples

```
genETMTree(n = 5)
```

genFordsAlphaTree *Generation of rooted binary trees under Ford's alpha model*

Description

genFordsAlphaTree - Generates a rooted binary tree in phylo format with the given number of n leaves under Ford's alpha model. Ford's alpha model is not a rate-based evolutionary (tree) construction and thus cannot generate edge lengths, only a topology. Instead, it works as follows: The idea is to start with a cherry and incrementally increase the size of the tree by adding a new leaf with a leaf edge to any edge (inner or leaf edge), one at a time. Given a tree with i leaves, then each of the i-1 innner edges (includes an additional root edge) is chosen with probability ALPHA/(i-ALPHA). Each of the i leaf edges is chosen with probability (1-ALPHA)/(i-ALPHA).
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genFordsAlphaTree(n, ALPHA)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
ALPHA	Numeric value >=0 and <=1 which specifies the probabilites of picking an inner or a leaf edge. For certain choices of ALPHA Ford's alpha model coincides with known models:

- ALPHA = 0: Yule model
- ALPHA = 1/2: PDA model (all phylogenies equally probable)
- ALPHA = 1: Caterpillar with n leaves

Value

genFordsAlphaTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- D. J. Ford. Probabilities on cladograms: introduction to the alpha model, 2005.
- G. Kaur, K. P. Choi, and T. Wu. Distributions of cherries and pitchforks for the Ford model. Theoretical Population Biology, 149:27–38, 2023.

Examples

```
genFordsAlphaTree(n = 5, ALPHA = 0.3)
```

---

genGFBTree *Generation of the greedy from the bottom tree*

Description

genGFBTree - Generates the rooted binary greedy from the bottom tree in phylo format with the given number of n leaves.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genGFBTree(n)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

Value

genGFBTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models. arXiv:2406.05185, 2024.
- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models: Supplementary material. https://tinyurl.com/278cwdh8, 2024.

Examples

```
genGFBTree(n = 6)
```

genGrowTree *Generation of rooted binary trees under tree growing models (no extinction)*

Description

genGrowTree - Generates a rooted binary tree in phylo format with the given number of n leaves under a specified discrete-time tree growing model without extinction. These tree growing models act at the leaves by varying their speciation rates according to a parameter ZETA or variance SIGMA. They may also depend on so-called trait values of the leaves (e.g., continuous or discrete age, or another numeric trait that affects fitness).
You may choose an already built-in model (see use_built_in) or specify a (new) model by defining how the rates (and optionally traits) change in every time step (see parameters childRates and otherRates as well as childTraits and otherTraits; see also Table 5 of the supplementary material of the corresponding manuscript).
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genGrowTree(
  n,
  STARTING_RATE = 1,
  STARTING_TRAIT = 10,
  ZETA = 1,
  SIGMA = 0,
  childRates,
  otherRates,
  childTraits = NULL,
  otherTraits = NULL,
  use_built_in = NULL
)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
STARTING_RATE	Positive numeric value (default = 1) which specifies the initial rate at which the speciation events occur (has only influence on the edge length, not on the tree topology).
STARTING_TRAIT	Numeric value (default = 10) which specifies the initial state of a trait.
ZETA	Constant non-negative numeric value (default = 1) which can influence the speciation rates. Can also be a vector if used as such when defining the functions childRates, otherRates, childTraits, and otherTraits.

SIGMA
: Constant positive numeric value (default = 0) which can influence the speciation rates. Can also be a vector if used as such when defining the functions childRates, otherRates, childTraits, and otherTraits.

childRates
: A function that generates two speciation rates for the children emerging from a speciation event based on various factors.
Necessary if use_built_in is not specified. childTraits works similarly but is executed before childRates.
All available parameters are:

  - the starting rate sr,
  - the starting trait value st,
  - the parent's rate pr,
  - the parent's trait value pt,
  - the children's trait values ct (vector ct[1] and ct[2]),
  - the parameters zeta ze
  - and sigma si.

  All parameters have to appear in the function definition but not necessarily in the body of the function. Trait values are NA, if childTraits and otherTraits is not given.
  Example:
  `function (sr, st, pr, pt, ct, ze, si) return(c(pr*ze,pr*(1-ze)))`
  for biased speciation.

otherRates
: A function that generates a new speciation rate for all leaves not affected by the speciation event (all but parent and children) based on various factors. The function is applied after the speciation event, i.e., after childRates/Traits.
Necessary if use_built_in is not specified. otherTraits works similarly.
All available parameters are:

  - the starting rate sr,
  - the starting trait value st,
  - the leaf's old rate or,
  - the leaf's old trait value ot,
  - the parameters zeta ze

- and sigma `si`.

  All parameters have to appear in the function definition but not necessarily in the body of the function. Trait values are NA, if `childTraits` and `otherTraits` is not given.
  Example:
  `function (sr, st, or, ot, ze, si) return(or*ze)` for age-step-based fertility.

`childTraits` An optional function (default = NULL) that generates two trait values for the children emerging from a speciation event based on various factors.
See `childRates` for available parameters (except `ct`) and explanations. Not necessary; is only applied if not NULL.
Example:
`function (sr, st, pr, pt, ze, si) return(c(0, 0))` for age.

`otherTraits` An optional function (default = NULL) that generates a new trait value for all leaves not affected by the speciation event (all but parent and children) based on various factors.
See `otherRates` for available parameters and explanations.
Not necessary; is only applied if not NULL.
Example:
`function (sr, st, or, ot, ze, si) return(ot+1)` for discrete age (age in time steps).

`use_built_in` Optional (default = NULL): Character specifying which of the already implemented models should be used. Overwrites `childRates`, `otherRates`, `childTraits`, and `otherTraits`.
Here is a list of available models with their (abbreviated) underlying functions given in parentheses (in order `childRates`, `otherRates`; then `childTraits` and `otherTraits` if necessary):

- "DCO_sym": Symmetric direct-children-only, `ZETA`>0 (c(sr ze, sr ze), sr)
- "DCO_asym": Asymmetric direct-children-only, `ZETA`>0 (c(sz, pr), sr)
- "IF_sym": Symmetric inherited fertility, `ZETA`>0 (c(pr ze, pr ze), or)
- "IF_asym": Asymmetric inherited fertility, `ZETA`>0 (c(pr ze, pr), or)
- "IF-diff": Unequal fertility inheritance, `ZETA`>=1 (c(2 pr ze / (ze+1), 2 pr / (ze+1)), or)
- "biased": Biased speciation, `ZETA` >=0 and <=1 (c(pr ze, pr (1-ze)), or)
- "ASB": Age-step-based fertility, `ZETA`>0 (c(sr, sr), or ze)
- "simpleBrown_sym": Symmetric simple Brownian, `SIGMA`> =0 (c(max{pr+ rnorm(1, mean=0, sd=si),1e-100}, max{pr+ rnorm(1, mean=0, sd=si),1e-

100}), or)

- "simpleBrown_asym": Asymmetric simple Brownian, SIGMA>=0 (c(max{pr+ rnorm(1, mean=0, sd=si),1e-100}, pr), or)

- "lin-Brown_sym": Sym. punctuated(-intermittent) linear-Brownian, SIGMA vector with two values >=0
(c(10^(log(ct[1])+ rnorm(1, mean=0, sd=si[1])), 10^(log(ct[2])+ rnorm(1, mean=0, sd=si[1]))), or;
c(max{pt + rnorm(1, mean=0, sd=si[2]),1e-100}, max{pt + rnorm(1, mean=0, sd=si[2]),1e-100}), ot)

- "lin-Brown_asym": Asym. punctuated(-intermittent) linear-Brownian, SIGMA vector with two values >=0
(c(10^(log(ct[1])+ rnorm(1, mean=0, sd=si[1])), pr), or;
c(max{pt + rnorm(1, mean=0, sd=si[2]),1e-100}, pt, ot)

- "lin-Brown-bounded_sym": Bounded sym. punctuated(-intermittent) linear-Brownian, SIGMA vector with two values >=0, STARTING_TRAIT is automatically set to 10
(c(10^(log(ct[1])+ rnorm(1, mean=0, sd=si[1])), 10^(log(ct[2])+ rnorm(1, mean=0, sd=si[1]))), or;
c(min{max{pt + rnorm(1, mean=0, sd=si[2]),1e-100},20}, min{max{pt + rnorm(1, mean=0, sd=si[2]),1e-100},20}), ot)

- "lin-Brown-bounded_asym": Bounded asym. punctuated(-intermittent) linear-Brownian, SIGMA vector with two values >=0
(c(10^(log(ct[1])+ rnorm(1, mean=0, sd=si[1])), pr), or;
c(min{max{pt + rnorm(1, mean=0, sd=si[2]),1e-100},20}, pt), ot)

- "log-Brown_sym": Sym. punctuated(-intermittent) log-Brownian, SIGMA vector with two values >=0
(c(10^(log(ct[1])+ rnorm(1, mean=0, sd=si[1])), 10^(log(ct[2])+ rnorm(1, mean=0, sd=si[1]))), or;
c(10^(log(pt)+ rnorm(1, mean=0, sd=si[2])), 10^(log(pt)+ rnorm(1, mean=0, sd=si[2]))), ot)

- "log-Brown_asym": Asym. punctuated(-intermittent) log-Brownian, SIGMA vector with two values >=0
(c(10^(log(ct[1])+ rnorm(1, mean=0, sd=si[1])), pr), or;
10^(c(log(pt)+ rnorm(1, mean=0, sd=si[2])), pt), ot)

Value

genGrowTree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models. arXiv:2406.05185, 2024.
- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models: Supplementary material. https://tinyurl.com/278cwdh8, 2024.
- M. G. B. Blum and O. Francois. On statistical tests of phylogenetic tree imbalance: the Sackin and other indices revisited. Mathematical Biosciences, 195(2):141–153, 2005.
- S. B. Heard. Patterns in phylogenetic tree balance with variable and evolving speciation rates. Evolution, 50(6):2141–2148, 1996.
- S. J. Kersting. Genetic programming as a means for generating improved tree balance indices (Master's thesis, University of Greifswald), 2020.
- M. Kirkpatrick and M. Slatkin. Searching for evolutionary patterns in the shape of a phylogenetic tree. Evolution, 47(4):1171–1181, 1993.

Examples

```
genGrowTree(n = 5, use_built_in = "IF_sym", ZETA = 2)
```

---

genMBTree *Generation of the maximally balanced tree*

Description

`genMBTree` - Generates the rooted binary maximally balanced tree in `phylo` format with the given number of `n` leaves.
Due to the restrictions of the `phylo` or `multiphylo` format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genMBTree(n)
```

Arguments

`n`	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

Value

`genMBTree` A single tree of class `phylo` is returned.

Author(s)

Sophie Kersting

References

- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models. arXiv:2406.05185, 2024.
- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models: Supplementary material. https://tinyurl.com/278cwdh8, 2024.

Examples

```
genMBTree(n = 6)
```

---

genPDATree *Generation of rooted binary trees under the PDA model*

Description

genPDATree - Generates a rooted binary tree in phylo format with the given number of n leaves under the proportional-to-distinguishable-arrangements model. Given n, all phylogenies (trees with labeled leaves) with n leaves are equiprobable under the PDA.
Due to the restrictions of the phylo or multiphylo format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genPDATree(n)
```

Arguments

n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

Value

genPDATree A single tree of class phylo is returned.

Author(s)

Sophie Kersting

References

- This function uses the `rtopology(..., rooted = T)` function of the `ape` package (E. Paradis, K. Schliep. "ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R." Bioinformatics, 35, 526-528, 2019).

- D. E. Rosen. Vicariant patterns and historical explanation in biogeography. Systematic Zoology, 27(2):159, 1978.

Examples

```
genPDATree(n = 5)
```

`genTrees` *Generation of rooted binary trees under a given tree model*

Description

`genTrees` - Is a wrapper function that generates `Ntrees`-many rooted binary trees with the given number of `n` leaves under any tree model `tm` contained in this package (more details on the available models are given in the parameter information for `tm`).
Due to the restrictions of the `phylo` or `multiPhylo` format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genTrees(n, Ntrees = 1L, tm)
```

Arguments

`n`	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
`Ntrees`	Integer value (default = 1) that specifies the desired number of generated trees.
`tm`	Character or list specifying the tree model under which the trees should be generated as well as their parameters. Available are:

- `"yule"` - Yule model.
- `"pda"` - PDA model.
- `"etm"` - ETM.
- `list("aldous", BETA)` - Aldous' beta splitting model with parameter `BETA` >= -2.

- `list("ford", ALPHA)` - Ford's alpha model with parameter `ALPHA` >= 0 and <= 1.

- `list("alt-birth-death", BIRTHRATE, DEATHRATE)` or `list("alt-birth-death", BIRTHRATE, DEATHRATE, TRIES)` - Alternative birth-death model with parameters `BIRTHRATE` >0 and `DEATHRATE`>= 0.

- `list("density", BIRTHRATE, EQUILIB)` or `list("density", BIRTHRATE, EQUILIB, TRIES, TIMEperTRY)` - Density dependent model with parameters `BIRTHRATE` >0 and `EQULIB`>= 1.

- `list("BiSSE", BIRTHRATES, DEATHRATES, TRANSRATES)` or `list("BiSSE", BIRTHRATES, DEATHRATES, TRANSRATES, TRIES, TIMEperTRY)` - BiSSE model with parameters `BIRTHRATES` (vector with 2 values >=0, one value >0), `DEATHRATES` (vector with 2 values >=0), and `TRANSRATES` (vector with 2 values >=0, one value >0).

- `list("DCO_sym", ZETA)` or `list("DCO_sym", ZETA, STARTING_RATE)` - Symmetric direct-children-only with parameter `ZETA` > 0 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("DCO_asym", ZETA)` or `list("DCO_asym", ZETA, STARTING_RATE)` - Asymmetric direct-children-only with parameter `ZETA` > 0 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("IF_sym", ZETA)` or `list("IF_sym", ZETA, STARTING_RATE)` - Symmetric inherited fertility with parameter `ZETA` > 0 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("IF_asym", ZETA)` or `list("IF_asym", ZETA, STARTING_RATE)` - Asymmetric inherited fertility with parameter `ZETA` > 0 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("IF-diff", ZETA)` or `list("IF-diff", ZETA, STARTING_RATE)` - Unequal fertility inheritance with parameter `ZETA` >= 1 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("biased", ZETA)` or `list("biased", ZETA, STARTING_RATE)` - Biased speciation with parameter `ZETA` >=0 and <=1 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("ASB", ZETA)` or `list("ASB", ZETA, STARTING_RATE)` - Age-step-based fertility with parameter `ZETA` > 0 and optionally `STARTING_RATE` > 0 (default = 1).

- `list("simpleBrown_sym", SIGMA)` or `list("simpleBrown_sym", SIGMA, STARTING_RATE)` - Symmetric simple Brownian with parameter `SIGMA` >=

0 and optionally STARTING_RATE > 0 (default = 1).

- list("simpleBrown_asym", SIGMA) or list("simpleBrown_asym", SIGMA, STARTING_RATE) - Asymmetric simple Brownian with parameter SIGMA >= 0 and optionally STARTING_RATE > 0 (default = 1).

- list("lin-Brown_sym", SIGMA) or list("lin-Brown_sym", SIGMA, STARTING_RATE, STARTING_TRAIT) - Sym. punctuated(-intermittent) linear-Brownian with parameter SIGMA (vector with 2 values >=0) and optionally STARTING_RATE > 0 (default = 1) and STARTING_TRAIT (default = 10).

- list("lin-Brown_asym", SIGMA) or list("lin-Brown_asym", SIGMA, STARTING_RATE, STARTING_TRAIT) - Asym. punctuated(-intermittent) linear-Brownian with parameter SIGMA (vector with 2 values >=0) and optionally STARTING_RATE > 0 (default = 1) and STARTING_TRAIT (default = 10).

- list("lin-Brown-bounded_sym", SIGMA) or list("lin-Brown-bounded_sym", SIGMA, STARTING_RATE, STARTING_TRAIT) - Bounded sym. punctuated(-intermittent) linear-Brownian with parameter SIGMA (vector with 2 values >=0) and optionally STARTING_RATE > 0 (default = 1) and STARTING_TRAIT (default = 10).

- list("lin-Brown-bounded_asym", SIGMA) or list("lin-Brown-bounded_asym", SIGMA, STARTING_RATE, STARTING_TRAIT) - Bounded asym. punctuated(-intermittent) linear-Brownian with parameter SIGMA (vector with 2 values >=0) and optionally STARTING_RATE > 0 (default = 1) and STARTING_TRAIT (default = 10).

- list("log-Brown_sym", SIGMA) or list("log-Brown_sym", SIGMA, STARTING_RATE, STARTING_TRAIT) - Sym. punctuated(-intermittent) log-Brownian with parameter SIGMA (vector with 2 values >=0) and optionally STARTING_RATE > 0 (default = 1) and STARTING_TRAIT (default = 10).

- list("log-Brown_asym", SIGMA) or list("log-Brown_asym", SIGMA, STARTING_RATE, STARTING_TRAIT) - Asym. punctuated(-intermittent) log-Brownian with parameter SIGMA (vector with 2 values >=0) and optionally STARTING_RATE > 0 (default = 1) and STARTING_TRAIT (default = 10).

  More information on each model and their parameters can be found in the description of each model, accessible with ?genYuleTree, ?genPDATree, ?genETMTree, ?genAldousBetaTree, ?genFordsAlphaTree, ?genBirthDeathTree, ?genAltBirthDeathTree, ?genGrowTree.

Value

genTrees If Ntrees is 1, then a single tree of class phylo is returned. If Ntrees is larger than 1, a list of class multiPhylo containing the trees of class phylo is returned.

Author(s)

Sophie Kersting

References

- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models. arXiv:2406.05185, 2024.
- S. J. Kersting, K. Wicke, and M. Fischer. Tree balance in phylogenetic models: Supplementary material. https://tinyurl.com/278cwdh8, 2024.

Examples

```
genTrees(n = 5, Ntrees = 2, tm = list("aldous", 1))
genTrees(n = 5, tm = "pda")
```

---

genYuleTree *Generation of rooted binary trees under the Yule model*

Description

`genYuleTree` - Generates a rooted binary tree in `phylo` format with the given number of `n` leaves under the Yule model. The Yule process is a simple birth-process in which all species have the same speciation rate.
Due to the restrictions of the `phylo` or `multiphylo` format the number of leaves must be at least 2 since there must be at least one edge.

Usage

```
genYuleTree(n)
```

Arguments

`n`	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

Value

`genYuleTree` A single tree of class `phylo` is returned.

Author(s)

Sophie Kersting

References

- This function uses the `rtree` function of the ape package (E. Paradis, K. Schliep. "ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R." Bioinformatics, 35, 526-528, 2019).

- G. U. Yule. A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis, F. R. S. Philosophical Transactions of the Royal Society of London. Series B, Containing Papers of a Biological Character, 213(402-410):21–87, 1925.

- E. F. Harding. The probabilities of rooted tree-shapes generated by random bifurcation. Advances in Applied Probability, 3(1):44–77, 1971.

Examples

```
genYuleTree(n = 5)
```

`getAccRegion` *Functions for computing the region of acceptance*

Description

`getAccRegion` - Computes the region of acceptance based on quantiles for a specified level of significance and method. The critical region is everything strictly outside this interval, i.e., the interval limits still belong to the acceptance region.

`getAccRegion_data` - Computes the region of acceptance for the given TSS values based on quantiles for a specified level of significance and method. The critical region is everything strictly outside this interval, i.e., the interval limits still belong to the acceptance region.

Usage

```
getAccRegion(
  tss,
  null_model = "yule",
  n,
  distribs = "sampled",
  sample_size = 1000L,
  test_type = "two-tailed",
  sig_lvl = 0.05
)

getAccRegion_data(tss_data, test_type = "two-tailed", sig_lvl = 0.05)
```

Arguments

tss	Vector containing the names (as character) of the tree shape statistics that should be compared. You may either use the short names provided in tssInfo to use the already included TSS, or use the name of a list object containing similar information as the entries in tssInfo. Example: Use "new_tss" as the name for the list object new_tss containing at least the function new_tss$func = function(tree){...}, and optionally also the information new_tss$short, new_tss$simple, new_tss$name, new_tss$type, new_tss$only_binary, and new_tss$safe_n.
null_model	The null model that is to be used to determine the power of the tree shape statistics. In general, it must be a function that produces rooted binary trees in phylo format. If the respective model is included in this package, then specify the model and its parameters by using a character or list. Available are all options listed under parameter tm in the documentation of function genTrees (type ?genTrees). If you want to include your own tree model, then use the name of a list object containing the function (with the two input parameters n and Ntrees). Example: Use "new_tm" for the list object new_tm <- list(func = function(n, Ntrees){...}).
n	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
distribs	Determines how the distributions and quantiles are computed. Available are: • "sampled" (default): sample_size many trees are sample under the respective models to determine the quantiles for the null model and how many trees under the alternative models lie outside.
sample_size	Sample size (integer >=10) if distributions are sampled (default= 1000L).
test_type	Determines the method. Available are: • "two-tailed" (default)
sig_lvl	Level of significance (default=0.05, must be >0 and <1).
tss_data	Numeric matrix of TSS values (one row per TSS). The row names are used as names for the TSS.

Value

getAccRegion Numeric matrix (one row per TSS) with two columns for the interval limits of the acceptance region.

getAccRegion_data Numeric matrix (one row per TSS) with two columns for the interval limits of the acceptance region.

Author(s)

Sophie Kersting

Examples

```
getAccRegion(tss = c("Sackin", "Colless", "B1I"), n = 8L, sample_size = 100L)
getAccRegion_data(getTSSdata(tss = c("Colless", "Sackin"), n = 8L,
                             Ntrees = 20L, tm = "yule"))
```

`getPowerMultAM` *Functions for computing the power*

Description

`getPowerMultAM` - Computes the power of a single TSS by calculating the proportion of values outside the region of acceptance for one or multiple alternative models.

`getPowerMultTSS` - Computes the power of one or multiple TSS by calculating the proportion of values outside the region of acceptance for a single alternative model.

Usage

```
getPowerMultAM(accept_region, alts_data)

getPowerMultTSS(accept_regions, alt_data)
```

Arguments

`accept_region`	Numeric vector of length two (increasing) setting the lower and upper limit of the region of acceptance (limits included).
`alts_data`	Numeric matrix with one column of values per alternative model. If there is only one alternative model, then it can be a simple vector of values instead (returns a single unnamed value).
`accept_regions`	Numeric matrix with two columns. Each column represents an interval, i.e., lower and upper limit of the region of acceptance (limits included), of a different TSS.
`alt_data`	Numeric matrix with one row of values under the alternative model per TSS. If there is only one TSS, then it can be a simple vector of values instead (returns a single unnamed value).

Value

`getPower` A vector containing the power regarding the given alternative models (retains column names of `alts_data`).

`getPowerMultTSS` A vector containing the power regarding the given TSS (retains row names of `accept_regions`).

Examples

```
getPowerMultAM(accept_region = c(10,20),
         alts_data = matrix(c(9,11,13,15,17,19,21,
                              5, 5, 5,15,25,25,25), ncol = 2, byrow = FALSE,
                            dimnames = list(NULL, c("AltTM1", "AltTM2"))))
getPowerMultAM(accept_region = c(10,20), alts_data = c(9,11,13,15,17,19,21))
getPowerMultTSS(accept_regions = matrix(c(10,20,
                                    110,120), ncol = 2, byrow = TRUE,
                            dimnames = list(c("TSS1", "TSS2"),NULL)),
         alt_data = matrix(c(  9, 14, 19, 24,
                              109,114,119,124), nrow = 2, byrow = TRUE,
                            dimnames = list(c("TSS1", "TSS2"),NULL)))
getPowerMultTSS(accept_regions = c(10,20), alt_data = c(9, 14, 19, 24))
```

`getTSSdata` *Functions for generating the TSS data under a tree model*

Description

`getTSSdata` - Compute the tree shape statistics of trees generated under a tree model for each given TSS.

`getTSSdata_trees` - Compute the tree shape statistics for each given TSS and all given trees.

Usage

```
getTSSdata(tss, n, Ntrees = 1L, tm)

getTSSdata_trees(tss, treeList)
```

Arguments

`tss`	Vector containing the names (as character) of the tree shape statistics that should be compared. You may either use the short names provided in `tssInfo` to use the already included TSS, or use the name of a list object containing similar information as the entries in `tssInfo`. Example: Use `"new_tss"` as the name for the list object `new_tss` containing at least the function `new_tss$func = function(tree){...}`, and optionally also the information `new_tss$short`, `new_tss$simple`, `new_tss$name`, `new_tss$type`, `new_tss$only_binary`, and `new_tss$safe_n`.
`n`	Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.
`Ntrees`	Integer value (default = 1) that specifies the desired number of generated trees.
`tm`	If the respective model is included in this package, then specify the model and its parameters by using a character or list. Available are all options listed under parameter `tm` in the documentation of function `genTrees` (type `?genTrees`). If you want to include your own tree model, then use the name of a list object containing the function (with the two input parameters `n` and `Ntrees`). Example: Use `"new_tm"` for the list object `new_tm <- list(func = function(n, Ntrees){...})`.

treeList List of trees of class multiphylo.

Value

getTSSdata Numeric matrix of TSS values (one row per TSS).

getTSSdata_trees Numeric matrix of TSS values (one row per TSS).

Author(s)

Sophie Kersting

Examples

```
# Example using tree models included in this package.
getTSSdata(tss = c("Colless", "Sackin"), n = 5L, Ntrees = 3L,
           tm = "yule")
# Example using a "new" TM provided by the user.
my_aldous <- list(func = function(n, Ntrees){
                        trees <- lapply(1:Ntrees,
                           function(x){genAldousBetaTree(n = n, BETA =5L)})
                        attr(trees, "class") <- "multiPhylo"
                        return(trees)})
getTSSdata(tss = c("Colless", "Sackin"), n = 5L, Ntrees = 3L,
           tm = "my_aldous")
# Example using TSS provided in tssInfo.
getTSSdata_trees(tss = c("Colless", "Sackin"),
           treeList = lapply(1:6L, function(x) genYuleTree(5)))
# Example using a "new" TSS provided by the user.
my_avd <- list(func = treebalance::avgVertDep, short = "My AVD")
getTSSdata_trees(tss = c("Colless", "my_avd"),
           treeList = lapply(1:6L, function(x) genYuleTree(5)))
```

---

getTSSnames *Get information on included tree shape statistics*

Description

getTSSnames - Returns the full names (character/expression) of the TSS.

getTSSsimple - Returns the simple names (character/expression) of the TSS.

getTSScolors - Returns the colors of the TSS.

getTSSsafe_n - Returns the ranges of n that can be safely used.

getTSStype - Returns the ranges of n that can be safely used.

getTSSonly_bin - Returns TRUE/FALSE vector: TRUE if TSS is only for binary trees and FALSE otherwise.

getAllTSS - Returns the short names of all TSS that are safe to use for the specified n, have one of the specified types and can be applied to (non-)binary trees (not_only_bin).

Usage

```
getTSSnames(tss_shorts)

getTSSsimple(tss_shorts)

getTSScolors(tss_shorts)

getTSSsafe_n(tss_shorts)

getTSStype(tss_shorts)

getTSSonly_bin(tss_shorts)

getAllTSS(n = NULL, not_only_bin = FALSE, types = c("tss", "bali", "imbali"))
```

Arguments

tss_shorts	Vector of short names (characters) of TSS contained in tssInfo.
n	Integer value or vector of integer values, that specifies the number(s) of leaves. If NULL (default), then getAllSafeTSS returns the short names of all TSS contained in tssInfo.
not_only_bin	Select TRUE if you also want to analyze non-binary trees and therefore want to filter out any TSS that only work on binary trees. Otherwise, select FALSE (default) if all TSS are applicable.
types	Character vector, that specifies all permissible TSS types. The vector may contain a subset of c("tss", "bali", "imbali") to indicate if balance indices, imbalance indices or mere TSS should be included. By default all types are permissible.

Value

getTSSnames Vector of characters/expressions.

getTSSsimple Vector of characters/expressions.

getTSScolors Vector of characters (color names).

getTSSsafe_n Numeric matrix, one row per TSS and two columns with lower and upper limit.

getTSStype Vector of characters (types as factors).

getTSSonly_bin Logical vector.

getAllTSS Character vector of short names of TSS contained in tssInfo.

Author(s)

Sophie Kersting

Examples

```
getTSSnames(tss_shorts = c("Sackin", "Colless", "B1I"))
getTSSsimple(tss_shorts = c("Sackin", "Colless", "B1I"))
getTSScolors(tss_shorts = c("Sackin", "Colless", "B1I"))
getTSSsafe_n(tss_shorts = c("Sackin", "Colless", "B1I"))
getTSStype(tss_shorts = c("Sackin", "Colless", "B1I"))
getTSSonly_bin(tss_shorts = c("Sackin", "Colless", "B1I"))
getAllTSS(n = c(3,30))
```

---

powerComp *Comparison of the power of TSS under different models*

Description

powerComp - Compare the power of a set of TSS to identify trees generated under different alternative models given a null model.

Usage

```
powerComp(
  tss,
  null_model = "yule",
  alt_models,
  n,
  distribs = "sampled",
  sample_size = 1000L,
  test_type = "two-tailed",
  sig_lvl = 0.05,
  with_CIradius = FALSE
)
```

Arguments

tss	Vector containing the names (as character) of the tree shape statistics that should be compared. You may either use the short names provided in tssInfo to use the already included TSS, or use the name of a list object containing similar information as the entries in tssInfo. Example: Use "new_tss" as the name for the list object new_tss containing at least the function new_tss$func = function(tree){...}, and optionally also the information new_tss$short, new_tss$simple, new_tss$name, new_tss$type, new_tss$only_binary, and new_tss$safe_n.
null_model	The null model that is to be used to determine the power of the tree shape statistics. In general, it must be a function that produces rooted binary trees in phylo format. If the respective model is included in this package, then specify the model and its parameters by using a character or list. Available are all options listed under parameter tm in the documentation of function genTrees (type ?genTrees).

If you want to include your own tree model, then use the name of a list object containing the function (dependent on one parameter n). Example:
Use "new_tm" for the list object
new_tm <- list(func = function(n, Ntrees){...}).

alt_models — List containing the alternative models that are to be used to determine the power of the tree shape statistics. Functions that produce rooted binary trees in phylo format. The information of each single model must be in the format described for null_model.

n — Integer value that specifies the desired number of leaves, i.e., vertices with in-degree 1 and out-degree 0.

distribs — Determines how the distributions and quantiles are computed. Available are:

- "sampled" (default): sample_size many trees are sample under the respective models to determine the quantiles for the null model and how many trees under the alternative models lie outside.

sample_size — Sample size (integer >=10) if distributions are sampled (default= 1000L).

test_type — Determines the method. Available are:

- "two-tailed" (default)

sig_lvl — Level of significance (default=0.05, must be >0 and <1).

with_CIradius — Determines if the radii for the 95%-confidence intervals of the power values should also be computed and returned (default = FALSE).

Value

powerComp Numeric matrix, one row per TSS and one column per alternative model if with_CIradius is FALSE (default). Else a list with two such numeric matrices, the first for the power and the second for the confidence interval radius.

Author(s)

Sophie Kersting

Examples

```
powerComp(tss = c("Sackin", "Colless", "B1I"),
          alt_models = list(list("aldous",-1), "pda", "etm"), n=10L,
          sample_size = 40L, with_CIradius = TRUE)
```

showTSSdata *Functions for displaying TSS distributions and TSS power*

Description

showTSSdata - This function plots histograms of TSS data.

showPower - This function displays the power of TSS under different alternative models.

showPower_param - This function displays the power of TSS under different representatives of a family of tree models which vary only in one parameter.

Usage

```
showTSSdata(
  tss_data,
  main = NULL,
  xlab = NULL,
  test_type = "two-tailed",
  sig_lvl = 0.05,
  ...
)

showPower(
  power_data,
  tss_names = NULL,
  tss_colors = NULL,
  model_names = NULL,
  sig_lvl = 0.05,
  ...
)

showPower_param(
  power_data,
  tss_names = NULL,
  tss_colors = NULL,
  tss_ltys = NULL,
  model_name = NULL,
  model_param,
  sig_lvl = 0.05,
  ...
)
```

Arguments

tss_data	Numeric matrix of TSS values (one row per TSS). The row names are used as names for the TSS.
main	Title (default = NULL). A generic title is created by default.

xlab	Label of x-axis (default = NULL). A generic label is created by default.
test_type	Determines the method. Available are: • "two-tailed" (default)
sig_lvl	Level of significance (default=0.05, must be >0 and <1). NO horizontal line is depicted if set to NULL.
...	Add further specifications for `plot()`.
power_data	Numeric matrix, one row per TSS and one column per alternative model.
tss_names	Vector of characters/expression of the TSS names (default = NULL). If none are provided, the row names of `power_data` are used as names for the TSS.
tss_colors	Vector of colors for the TSS (default = NULL).
model_names	Vector of characters/expression of the model names (default = NULL). If none are provided, the column names of `power_data` are used as names for the models.
tss_ltys	Vector of line types for the TSS (default = NULL).
model_name	Vector of characters/expressions of the name of the tree model family and of the parameter (default = NULL), e.g. `c("Aldous\'", expression(beta))`. If none are provided, the first column name of `power_data` is used.
model_param	Numeric vector containing the parameter values of the representatives of the tree model.

Value

`showTSSdata` No return value, called for side effects (plotting).

`showPower` No return value, called for side effects (plotting).

`showPower_param` No return value, called for side effects (plotting).

Author(s)

Sophie Kersting

Examples

```
showTSSdata(tss_data = getTSSdata_trees(tss = c("Colless", "Sackin"),
                treeList = lapply(1:20L, function(x) genYuleTree(10))),
                breaks=15)
# With error bars:
showPower(power_data = powerComp(tss = c("Sackin", "Colless", "B1I"),
                                    alt_models = list("pda", "etm"), n=10L,
                                    sample_size = 50L,
                                    with_CIradius = TRUE),
              tss_names = getTSSnames(c("Sackin", "Colless", "B1I")),
              tss_colors = getTSScolors(c("Sackin", "Colless", "B1I")),
              model_names = c("PDA", "ETM"),
              main = "Power (Yule as null model, n = 10, N=50)",
              ylim = c(0,1), ylab = "Power (null model rejected)")
```

```
# Without error bars:
showPower(power_data = powerComp(tss = c("Sackin", "Colless", "B1I"),
                                 alt_models = list("pda", "etm"), n=10L,
                                 sample_size = 50L),
              tss_names = getTSSnames(c("Sackin", "Colless", "B1I")),
              tss_colors = getTSScolors(c("Sackin", "Colless", "B1I")),
              model_names = c("PDA", "ETM"),
              main = "Power (Yule as null model, n = 10, N=50)",
              ylim = c(0,1), ylab = "Power (null model rejected)")
# With confidence bands:
showPower_param(power_data = powerComp(tss = c("Sackin", "Colless", "B1I"),
                                 alt_models = list(list("aldous", 0.5),
                                 list("aldous", 0),list("aldous", -0.5),
                                 list("aldous", -1),list("aldous", -1.5)),
                                 n=20L, sample_size = 50L,
                                 with_CIradius = TRUE),
              tss_names = getTSSnames(c("Sackin", "Colless", "B1I")),
              tss_colors = getTSScolors(c("Sackin", "Colless", "B1I")),
              model_name = c("Aldous-beta splitting model", "beta"),
              model_param = c(0.5,0,-0.5,-1,-1.5),
              ylim = c(0,1), ylab = "Power (null model rejected)")
# Without confidence bands:
showPower_param(power_data = powerComp(tss = c("Sackin", "Colless", "B1I"),
                                 alt_models = list(list("aldous", 0.5),
                                 list("aldous", 0),list("aldous", -0.5),
                                 list("aldous", -1),list("aldous", -1.5)),
                                 n=20L, sample_size = 50L),
              tss_names = getTSSnames(c("Sackin", "Colless", "B1I")),
              tss_colors = getTSScolors(c("Sackin", "Colless", "B1I")),
              model_name = c("Aldous-beta splitting model", "beta"),
              model_param = c(0.5,0,-0.5,-1,-1.5),
              ylim = c(0,1), ylab = "Power (null model rejected)")
```

`tssInfo` *Tree shape statistics*

Description

`tssInfo` - List that provides information on available tree shape statistics (TSS) from the package 'treebalance'. Most of them are either balance or imbalance indices. The indices are grouped by their families and otherwise sorted alphabetically by their full names.
The following information is provided:

- short: Abbreviation of the name (plain characters).
- simple: Simplified full name (plain characters).
- name: Full name (partly expressions as some names use special symbols).

- func: Function of the TSS.
- type: Either "tss", "bali", or "imbali" expressing what type of tree shape statistic it is.
- only_binary: TRUE if TSS is suitable only for binary trees, FALSE if also applicable to arbitrary rooted trees.
- safe_n : Integer vector with two entries specifying the range of leaf numbers `n` for which the TSS can be (safely) used, without warnings for too few leaves or values reaching Inf for too many leaves.
  c(4,800), for example means that this TSS should only be applied on trees with 4 to 800 leaves.
  'Inf' as the second entry means that there is no specific upper limit, but that the size of the tree itself and the computation time are the limiting factors.
- col: Color for the TSS (related TSS have similar colors).

Usage

```
tssInfo
```

Format

An object of class `list` of length 29.

Author(s)

Sophie Kersting

References

- M. Fischer, L.Herbst, S. J. Kersting, L. Kühn, and K. Wicke, Tree Balance Indices - A Comprehensive Survey. Springer, 2023. ISBN: 978-3-031-39799-8

Examples

```
tssInfo$ALD$name
tssInfo$ALD$func(genYuleTree(6))
```

Index