Package 'ddtlcm'

September 14, 2023

Type Package

Title Latent Class Analysis with Dirichlet Diffusion Tree Process Prior

Version 0.1.1

Date 2023-08-26

Maintainer Mengbing Li <mengbing@umich.edu>

Description Implements a Bayesian algorithm to fit latent class models, particularly useful for weakly separated latent classes. Reference: Li et al. (2023) <arXiv:2306.04700>.

Depends R(>= 4.0.0)

Imports ape (>= 5.6-2), data.table (>= 1.14.4), extraDistr (>= 1.9.1), ggplot2 (>= 3.4.0), ggpubr (>= 0.6.0), ggtext (>= 0.1.2), ggtree (>= 3.4.0), label.switching (>= 1.8), matrixStats (>= 0.62.0), methods (>= 4.2.3), phylobase (>= 0.8.10), poLCA (>= 1.6.0.1), truncnorm (>= 1.0-8), BayesLogit (>= 2.1), Matrix (>= 1.5-1), Rdpack (>= 2.5), R.utils (>= 2.12.2)

Suggests knitr, parallel, testthat, rmarkdown, xfun

License MIT + file LICENSE

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.2.3

URL https://github.com/limengbinggz/ddtlcm

BugReports https://github.com/limengbinggz/ddtlcm/issues

NeedsCompilation no

Author Mengbing Li [cre, aut] (<https://orcid.org/0000-0002-2264-8006>), Briana Stephenson [ctb], Zhenke Wu [ctb]

Repository CRAN

Date/Publication 2023-09-14 19:10:02 UTC

R topics documented:

`add_leaf_branch` *Add a leaf branch to an existing tree tree_old*

Description

Add a leaf branch to an existing tree tree_old

Usage

```
add_leaf_branch(tree_old, div_t, new_leaf_label, where, position)
```

Arguments

`tree_old`	the original "phylo" tree (with K leaves) to which the leaf branch will be added
`div_t`	divergence time of the new branch
`new_leaf_label`	the label of the newly added leaf
`where`	node name of to which node in the existing tree the new leaf branch should connect to
`position`	the numerical location of the left side of the added branch

Value

a "phylo" tree with K+1 leaves

`add_multichotomous_tip` *Add a leaf branch to an existing tree tree_old to make a multichotomus branch*

Description

Add a leaf branch to an existing tree tree_old to make a multichotomus branch

Usage

```
add_multichotomous_tip(tree_old, div_t, new_leaf_label, where)
```

Arguments

`tree_old`	the original "phylo" tree (with K leaves) to which the leaf branch will be added
`div_t`	divergence time of the new branch
`new_leaf_label`	the label of the newly added leaf
`where`	node name of to which node in the existing tree the new leaf branch should connect to

Value

a "phylo" tree with K+1 leaves that could possibly be multichotomus

`add_one_sample` *Functions to simulate trees and node parameters from a DDT process. Add a branch to an existing tree according to the branching process of DDT*

Description

Functions to simulate trees and node parameters from a DDT process. Add a branch to an existing tree according to the branching process of DDT

Usage

```
add_one_sample(tree_old, c, c_order, theta, alpha)
```

Arguments

tree_old	a "phylo" object. The tree (K leaves) to which a new branch will be added.
c	hyparameter of divergence function a(t)
c_order	equals 1 (default) or 2 to choose divergence function a(t) = c/(1-t) or c/(1-t)^2.
alpha, theta	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) Allowable range: 0 <= beta <= 1, and alpha >= -2 beta For DDT, alpha = theta = 0. For general multifurcating tree from a Pitman-Yor process, specify positive values to alpha and theta. It is, however, recommended using alpha = theta = 0 in inference because multifurcating trees have not been tested rigorously.

Value

a "phylo" object. A tree with K+1 leaves. if t2 > t1, then select which path to take, with probability proportional to the number of data points that already traversed the path

---

add_root *Add a singular root node to an existing nonsingular tree*

Description

Add a singular root node to an existing nonsingular tree

Usage

```
add_root(tree_old, root_edge_length, root_label, leaf_label)
```

Arguments

tree_old	the original nonsingular "phylo" tree
root_edge_length	a number in (0, 1) representing the distance between the new and the original root nodes
root_label	a character label of the new root node
leaf_label	a character label of the leaf node

Value

a singular "phylo" tree

`attach_subtree` *Attach a subtree to a given DDT at a randomly selected location*

Description

Attach a subtree to a given DDT at a randomly selected location

Usage

```
attach_subtree(
  subtree,
  tree_kept,
  detach_div_time,
  pa_detach_node_label,
  c,
  c_order = 1,
  theta = 0,
  alpha = 0
)
```

Arguments

`subtree`	subtree to attach to tree_kept
`tree_kept`	the tree to be attached to
`detach_div_time`	divergence time of subtree when it was extracted from the original tree
`pa_detach_node_label`	label of the parent node of the detached node
`c`	hyparameter of divergence function a(t)
`c_order`	equals 1 (default) or 2 to choose divergence function
`alpha, theta`	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) For DDT, alpha = theta = 0. For general multifurcating tree from a Pitman-Yor process, specify positive values to alpha and theta. It is, however, recommended using alpha = theta = 0 in inference because multifurcating trees have not been tested rigorously.

See Also

Other sample trees: `random_detach_subtree()`, `reattach_point()`

a_t_one *Compute divergence function*

Description

Compute value, cumulative hazard, and inverse for divergence function a(t) = c / (1-t)

Usage

```
a_t_one(c, t)

a_t_one_cum(c, t)

A_t_inv_one(c, y)
```

Arguments

c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
t	a number in the interval (0, 1) indicating the divergence time
y	a positive number to take inverse

Value

The value and cumulative hazard return a positive number. The inverse function returns a number in the interval (0, 1).

Functions

- `a_t_one()`: value of the divergence function
- `a_t_one_cum()`: cumulative hazard function
- `A_t_inv_one()`: inverse function

See Also

Other divergence functions: `a_t_two()`

Examples

```
a_t_one(1, 0.5)
a_t_one_cum(1, 0.5)
A_t_inv_one(1, 2)
```

a_t_two *Compute divergence function*

Description

Compute value, cumulative hazard, and inverse for divergence function a(t) = c / (1-t)^2

Usage

```
a_t_two(c, t)

a_t_two_cum(c, t)

A_t_inv_two(c, y)
```

Arguments

c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
t	a number in the interval (0, 1) indicating the divergence time
y	a positive number to take inverse

Value

The value and cumulative hazard return a positive number. The inverse function returns a number in the interval (0, 1).

Functions

- `a_t_two()`: value of the divergence function
- `a_t_two_cum()`: cumulative hazard function
- `A_t_inv_two()`: inverse function

See Also

Other divergence functions: `a_t_one()`

Examples

```
a_t_two(1, 0.5)
a_t_two_cum(1, 0.5)
A_t_inv_two(1, 2)
```

compute_IC *Compute information criteria for the DDT-LCM model*

Description

Compute information criteria for the DDT-LCM model, including the Widely Applicable Information Criterion (WAIC), and Deviance Information Criterion (DIC). WAIC and DIC are computed using two different methods described in Gelman, Hwang, and Vehtari (2013), one based on (1) posterior means and the other based on (2) posterior variances.

Usage

```
compute_IC(result, burnin = 5000, ncores = 1L)
```

Arguments

result	a "ddt_lcm" object
burnin	an integer specifying the number of burn-in iterations from MCMC chain
ncores	an integer specifying the number of cores to compute marginal posterior log-likelihood in parallel

Value

a named list of the following elements

WAIC_result a list of WAIC-related results computed using the two methods

DIC1 DIC computed using method 1.

DIC2 DIC computed using method 2.

Examples

```
data(result_hchs)
IC_result <- compute_IC(result = result_hchs, burnin = 50, ncores = 1L)
```

create_leaf_cor_matrix *Create a tree-structured covariance matrix from a given tree*

Description

Retrieve the covariance matrix of leaf nodes of a DDT tree

Usage

```
create_leaf_cor_matrix(tree_phylo4d)
```

Arguments

`tree_phylo4d`	a "phylo4d" object

Value

a K by K covariance matrix

Examples

```
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_synthetic)
# extract elements into the global environment
list2env(setNames(data_synthetic, names(data_synthetic)), envir = globalenv())
create_leaf_cor_matrix(tree_with_parameter)
```

`data_hchs` *Parameters for the HCHS dietary recall data example*

Description

A list of five variables containing the food items and parameter estimates obtained from MCMC chains. The real multivariate binary dataset is not included for privacy. The variables are as follows:

Usage

```
data(data_hchs)
```

Format

An object of class `list` of length 5.

Details

- item_membership_list. A list of G = 7 elements, where the g-th element contains the indices of items belonging to major food group g.
- item_name_list. A list of G = 7 elements, where the g-th element contains the item labels of items in major food group g, and the name of the g-th element is the major food group label.
- tree_phylo. The maximum a posterior tree estimate with K = 6 leaves obtained from the real HCHS data. Class "phylo".
- class_probability. A K-vector with entries between 0 and 1. The posterior mean estimate for class probabilities obtained from the real HCHS data.
- Sigma_by_group. A G-vector greater than 0. The posterior mean estimate for group-specific diffusion variances obtained from the real HCHS data.

References

https://arxiv.org/abs/2306.04700

data_synthetic *Synthetic data example*

Description

This list contains one synthetic data with K = 3 latent classes. The elements are as follows:

Usage

```
data(data_synthetic)
```

Format

A list with 8 elements

Details

- tree_phylo. A "phylo" tree with K = 3 leaves.
- class_probability. A K-vector with entries between 0 and 1.
- item_membership_list. A list of G = 7 elements, where the g-th element contains the indices of items belonging to major food group g.
- item_name_list. A list of G = 7 elements, where the g-th element contains the item labels of items in major food group g, and the name of the g-th element is the major food group label.
- Sigma_by_group. A G-vector greater than 0. The group-specific diffusion variances.
- response_matrix. A binary matrix with N = 100 rows and J = 80 columns. Each row contains a multivariate binary response vector of a synthetic individual.
- response_prob. A K by J probability matrix. The k-th row contains the item response probabilities of class k.
- tree_with_parameter. A `phylobase::phylo4d` object. Basically the tree_phylo embedded with additional logit(response_prob) at the leaf nodes.

ddtlcm *Dirichlet diffusion tree-latent class model (DDT-LCM)*

Description

`ddtlcm` is designed for clustering multivariate binary observations over grouped items while (1) leveraging between-cluster similarities guided by an unknown tree that is simultaneously estimated, and (2) accounting for varying degrees of shrinkage across major item groups. Classes positioned closer on the tree exhibit more similarities a priori. The model guards against potential numerical and statistical instability of classical LCMs especially when classes are weakly separated under small sample sizes. This is achieved by equipping a LCM with a DDT process prior on the class profiles, which are the class-conditional response probabilities. The posterior inference algorithm is based on Metropolis-Hastings algorithm for sampling the tree structure, and Gibbs sampler with Polya-Gamma augmentation for the LCM parameters.

main ddtlcm wrapper function

`ddtlcm_fit()`

See Also

- https://github.com/limengbinggz/ddtlcm for the source code and system/software requirements to use `ddtlcm` for your data.

`ddtlcm_fit` *MH-within-Gibbs sampler to sample from the full posterior distribution of DDT-LCM*

Description

Use DDT-LCM to estimate latent class and tree on class profiles for multivariate binary outcomes.

Usage

```
ddtlcm_fit(
  K,
  data,
  item_membership_list,
  total_iters = 5000,
  initials = list(),
  priors = list(),
  controls = list(),
 initialize_args = list(method_lcm = "random", method_dist = "euclidean", method_hclust
    = "ward.D", method_add_root = "min_cor", alpha = 0, theta = 0)
)
```

Arguments

`K`	number of classes (integer)
`data`	an NxJ matrix of multivariate binary responses, where N is the number of individuals, and J is the number of granular items
`item_membership_list`	a list of G elements, where the g-th element contains the column indices of `data` corresponding to items in major group g, and G is number of major item groups
`total_iters`	number of posterior samples to collect (integer)
`initials`	a named list of initial values of the following parameters:
	`tree_phylo4d` a phylo4d object. The initial tree have K leaves (labeled as "v1" through "vK"), 1 singleton root node (labeled as "u1"), and K-1 internal nodes (labeled as "u1" through "uK-1"). The tree also contains parameters for the leaf nodes and the root node (which equals 0). The parameters for the internal nodes can be NAs because they will not be used in the algorithm.

response_prob a K by J matrix with entries between 0 and 1. The initial values for the item response probabilities. They should equal to the expit-transformed leaf parameters of tree_phylo4d.

class_probability a K-vector with entries between 0 and 1. The initial values for the class probabilities. Entries should be nonzero and sum up to 1, or otherwise will be normalized

class_assignments a N-vector with integer entries from 1, ..., K. The initial values for individual class assignments.

Sigma_by_group a G-vector greater than 0. The initial values for the group-specific diffusion variances.

c a value greater than 0. The initial values for the group-specific diffusion variances.

Parameters not supplied with initial values will be initialized using the initialize function with arguments in initialize_args.

priors a named list of values of hyperparameters of priors. See the function initialize for explanation.

shape_sigma a G-vector of positive values. The g-th element is the shape parameter for the inverse-Gamma prior on diffusion variance parameter sigma_g^2. Default is rep(2, G).

rate_sigma a G-vector of positive values. Rate parameter. See above. Default is rep(2, G).

prior_dirichlet a K-vector with entries positive entries. The parameter of the Dirichlet prior on class probability.

shape_c a positive value. The shape parameter for the Gamma prior on divergence function hyperparameter c. Default is 1.

rate_c a positive value. The rate parameter for c. Default is 1.

a_pg a positive value. The scale parameter for the generalized logistic distribution used in the augmented Gibbs sampler for leaf parameters. Default is 1, corresponding to the standard logistic distribution.

controls a named list of control variables.

fix_tree a logical. If TRUE (default), the tree structure will be sampled in the algorithm. If FALSE, the tree structure will be fixed at the initial input.

c_order a numeric value. If 1, the divergence function is a(t) = c/(1-t). If 2, the divergence function is a(t) = c/(1-t)^2.

initialize_args

a named list of initialization arguments. See the function initialize for explanation.

Value

an object of class "ddt_lcm"; a list containing the following elements:

tree_samples a list of information of the tree collected from the sampling algorithm, including:
accept: a binary vector where 1 indicates acceptance of the proposal tree and 0 indicates

rejection. tree_list: a list of posterior samples of the tree. dist_mat_list: a list of tree-structured covariance matrices representing the marginal covariances among the leaf parameters, integrating out the internal node parameters and all intermediate stochastic paths in the DDT branching process.

response_probs_samples a total_iters x K x J array of posterior samples of item response probabilities

class_probs_samples a K x total_iters matrix of posterior samples of class probabilities

Z_samples a N x total_iters integer matrix of posterior samples of individual class assignments

Sigma_by_group_samples a G x total_iters matrix of posterior samples of diffusion variances

c_samples a total_iters vector of posterior samples of divergence function hyperparameter

loglikelihood a total_iters vector of log-likelihoods of the full model

loglikelihood_lcm a total_iters vector of log-likelihoods of the LCM model only

setting a list of model setup information, including: K, item_membership_list, and G

controls a list of model controls, including: fix_tree: FALSE to perform MH sampling of the tree, TRUE to fix the tree at the initial input. c_order: a numeric value of 1 or 2 (see Arguments))

data the input data matrix

Examples

```
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_synthetic)
# extract elements into the global environment
list2env(setNames(data_synthetic, names(data_synthetic)), envir = globalenv())
# run DDT-LCM
result <- ddtlcm_fit(K = 3, data = response_matrix, item_membership_list, total_iters = 50)
```

---

div_time *Sample divergence time on an edge uv previously traversed by m(v) data points*

Description

Sample divergence time on an edge uv previously traversed by m(v) data points

Usage

```
div_time(t_u, m_v, c, c_order = 1, alpha = 0, theta = 0)
```

Arguments

t_u	a number in the interval (0, 1) indicating the divergence time at node u
m_v	an integer for the number of data points traversed through node v
c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
c_order	equals 1 if using divergence function a(t) = c / (1-t), or 2 if a(t) = c / (1-t)^2. Default is 1
alpha, theta	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) For DDT, alpha = theta = 0. For general multifurcating tree from a Pitman-Yor process, specify positive values to alpha and theta. It is, however, recommended using alpha = theta = 0 in inference because multifurcating trees have not been tested rigorously.

Value

a number in the interval (0, 1)

---

draw_mnorm *Efficiently sample multivariate normal using precision matrix from x ~ N(Q^-1a, Q^-1), where Q^-1 is the precision matrix*

Description

Efficiently sample multivariate normal using precision matrix from x ~ N(Q^-1a, Q^-1), where Q^-1 is the precision matrix

Usage

```
draw_mnorm(precision_mat, precision_a_vec)
```

Arguments

precision_mat	precision matrix Q of the multivariate normal distribution
precision_a_vec	a vector a such that the mean of the multivariate normal distribution is Q^-1a

expit *The expit function*

Description

The expit function: f(x) = exp(x) / (1+exp(x)), computed in a way to avoid numerical underflow.

Usage

```
expit(x)
```

Arguments

x	a value or a numeric vector between 0 and 1 (exclusive)

Value

a number or real-valued vector

Examples

```
expit(0.2)
expit(c(-1, -0.3, 0.6))
```

exp_normalize *Compute normalized probabilities: exp(x_i) / sum_j exp(x_j)*

Description

Compute normalized probabilities: exp(x_i) / sum_j exp(x_j)

Usage

```
exp_normalize(x)
```

Arguments

x	a number or rea-valued vector

Value

a number or rea-valued vector

H_n *Harmonic series*

Description

Harmonic series

Usage

```
H_n(k)
```

Arguments

k	a positive integer

initialize *Initialize the MH-within-Gibbs algorithm for DDT-LCM*

Description

Initialize the MH-within-Gibbs algorithm for DDT-LCM

Usage

```
initialize(
  K,
  data,
  item_membership_list,
  c = 1,
  c_order = 1,
  method_lcm = "random",
  method_dist = "euclidean",
  method_hclust = "ward.D",
  method_add_root = "min_cor",
  fixed_initials = list(),
  fixed_priors = list(),
  alpha = 0,
  theta = 0,
  ...
)
```

Arguments

K	number of classes (integer)
data	an NxJ matrix of multivariate binary responses, where N is the number of individuals, and J is the number of granular items
item_membership_list	a list of G elements, where the g-th element contains the column indices of data corresponding to items in major group g
c	hyparameter of divergence function a(t)
c_order	equals 1 (default) or 2 to choose divergence function a(t) = c/(1-t) or c/(1-t)^2.
method_lcm	a character. If random (default), the initial LCM parameters will be random values. If poLCA, the initial LCM parameters will be EM algorithm estimates from the poLCA function.
method_dist	string specifying the distance measure to be used in dist(). This must be one of "euclidean" (defaults), "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.
method_hclust	string specifying the distance measure to be used in hclust(). This should be (an unambiguous abbreviation of) one of "ward.D" (defaults), "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
method_add_root	string specifying the method to add the initial branch to the tree output from hclust(). This should be one of "min_cor" (the absolute value of the minimum between-class correlation; default) or "sample_ddt" (randomly sample a small divergence time from the DDT process with c = 100)
fixed_initials	a named list of fixed initial values, including the initial values for tree ("phylo4d"), response_prob, class_probability, class_assignments, Sigma_by_group, and c. Default is NULL. See
fixed_priors	a named list of fixed prior hyperparameters, including the the Gamma prior for c, inverse-Gamma prior for sigma_g^2, and Dirichlet prior for pi. Moreover, we allow for a type III generalized logistic distribution such that f(eta; a_pg) = theta. This becomes a standard logistic distribution when a_pg = 1. See Dalla Valle, L., Leisen, F., Rossini, L., & Zhu, W. (2021). A Pólya–Gamma sampler for a generalized logistic regression. Journal of Statistical Computation and Simulation, 91(14), 2899-2916. An example input list is list("shape_c" = 1, "rate_c" = 1, "shape_sigma" = rep(2, G), "rate_sigma" = rep(2, G), "a_pg" = 1.0), where G is the number of major item groups. Default is NULL.
alpha, theta	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) For DDT, alpha = theta = 0
...	optional arguments for the poLCA function

Value

phylo4d object of tree topology

See Also

ddtlcm_fit()

Other initialization functions: initialize_hclust(), initialize_poLCA()

Examples

```
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_synthetic)
# extract elements into the global environment
list2env(setNames(data_synthetic, names(data_synthetic)), envir = globalenv())
K <- 3
G <- length(item_membership_list)
fixed_initials <- list("shape_c" = 2, "rate_c" = 2)
fixed_priors <- list("rate_sigma" = rep(3, G))
initials <- initialize(K, data = response_matrix, item_membership_list,
  c=1, c_order=1, fixed_initials = fixed_initials, fixed_priors = fixed_priors)
```

---

initialize_hclust *Estimate an initial binary tree on latent classes using hclust()*

Description

Estimate an initial binary tree on latent classes using hclust()

Usage

```
initialize_hclust(
  leaf_data,
  c,
  c_order = 1,
  method_dist = "euclidean",
  method_hclust = "ward.D",
  method_add_root = "min_cor",
  alpha = 0,
  theta = 0,
  ...
)
```

Arguments

leaf_data	a K by J matrix of logit(theta_kj)
c	hyparameter of divergence function a(t)
c_order	equals 1 (default) or 2 to choose divergence function
method_dist	string specifying the distance measure to be used in dist(). This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski". Any unambiguous substring can be given.

method_hclust	string specifying the distance measure to be used in hclust(). This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
method_add_root	string specifying the method to add the initial branch to the tree output from hclust(). This should be one of "min_cor" (the absolute value of the minimum between-class correlation) or "sample_ddt" (randomly sample a small divergence time from the DDT process with a large c = 100)
alpha, theta	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) For DDT, alpha = theta = 0
...	optional arguments for the poLCA function

Value

phylo4d object of tree topology

See Also

Other initialization functions: initialize_poLCA(), initialize()

---

initialize_poLCA *Estimate an initial response profile from latent class model using poLCA()*

Description

Estimate an initial response profile from latent class model using poLCA()

Usage

```
initialize_poLCA(K, data, ...)
```

Arguments

K	number of latent classes
data	a N by J observed binary matrix, where the i,j-th element is the response of item j for individual i
...	optional arguments for the poLCA function

Value

a K by J probability matrix, the k,j-th entry being the response probability to item j of an individual in class k

See Also

Other initialization functions: initialize_hclust(), initialize()

initialize_randomLCM *Provide a random initial response profile based on latent class mode*

Description

Provide a random initial response profile based on latent class mode

Usage

```
initialize_randomLCM(K, data)
```

Arguments

K	number of latent classes
data	a N by J observed binary matrix, where the i,j-th element is the response of item j for individual i

Value

a K by J probability matrix, the k,j-th entry being the response probability to item j of an individual in class k

J_n *Compute factor in the exponent of the divergence time distribution*

Description

Compute factor in the exponent of the divergence time distribution

Usage

```
J_n(l, r)
```

Arguments

l	number of data points to the left
r	number of data points to the right

logit *The logistic function*

Description

The logit function: f(x) = log(x / (1/x)). Large absolute values of x will be truncated to +/- 5 after logit transformation according to its sign.

Usage

```
logit(x)
```

Arguments

x	a value or a numeric vector between 0 and 1 (exclusive)

Value

a number or rea-valued vector

Examples

```
logit(0.2)
logit(c(0.2, 0.6, 0.95))
```

logllk_ddt *Calculate loglikelihood of a DDT, including the tree structure and node parameters*

Description

Calculate loglikelihood of a DDT, including the tree structure and node parameters

Usage

```
logllk_ddt(
  c,
  c_order,
  Sigma_by_group,
  tree_phylo4d,
  item_membership_list,
  tree_structure_old = NULL,
  dist_mat_old = NULL
)
```

Arguments

c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
c_order	equals 1 if using divergence function a(t) = c / (1-t), or 2 if a(t) = c / (1-t)^2. Default is 1
Sigma_by_group	a vector of diffusion variances of G groups from the previous iteration
tree_phylo4d	a "phylo4d" object
item_membership_list	a list of G elements, where the g-th element contains the column indices of data corresponding to items in major group g
tree_structure_old	a list of at least named elements: loglikelihoods of the input tree topology and divergence times. These can be directly obtained from the return of this function. Default is NULL. If given a list, then computation of the loglikelihoods will be skipped to save time. This is useful in the Metropolis-Hasting algorithm when the previous proposal is not accepted.
dist_mat_old	a tree-structured covariance matrix from a given tree. Default is NULL.

Value

a numeric of loglikelihood

See Also

Other likelihood functions: logllk_ddt_lcm(), logllk_div_time_one(), logllk_div_time_two(), logllk_lcm(), logllk_location(), logllk_tree_topology()

---

logllk_ddt_lcm *Calculate loglikelihood of the DDT-LCM*

Description

Calculate loglikelihood of the DDT-LCM

Usage

```
logllk_ddt_lcm(
  c,
  Sigma_by_group,
  tree_phylo4d,
  item_membership_list,
  tree_structure_old = NULL,
  dist_mat_old = NULL,
  response_matrix,
  leaf_data,
```

```
  prior_class_probability,
  prior_dirichlet,
  ClassItem,
  Class_count
)
```

Arguments

c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
Sigma_by_group	a vector of diffusion variances of G groups
tree_phylo4d	a "phylo4d" object
item_membership_list	a list of G elements, where the g-th element contains the column indices of data corresponding to items in major group g
tree_structure_old	a list of at least named elements: loglikelihoods of the input tree topology and divergence times. These can be directly obtained from the return of this function. Default is NULL. If given a list, then computation of the loglikelihoods will be skipped to save time. This is useful in the Metropolis-Hasting algorithm when the previous proposal is not accepted.
dist_mat_old	a tree-structured covariance matrix from a given tree. Default is NULL.
response_matrix	a N by J binary matrix, where the i,j-th element is the response of item j for individual i
leaf_data	a K by J matrix of logit(theta_kj)
prior_class_probability	a length K vector, where the k-th element is the probability of assigning an individual to class k. It does not have to sum up to 1
prior_dirichlet	a vector of length K. The Dirichlet prior of class probabilities
ClassItem	a K by J matrix, where the k,j-th element counts the number of individuals that belong to class k have a positive response to item j
Class_count	a length K vector, where the k-th element counts the number of individuals belonging to class k

Value

a numeric of loglikelihood

See Also

Other likelihood functions: logllk_ddt(), logllk_div_time_one(), logllk_div_time_two(), logllk_lcm(), logllk_location(), logllk_tree_topology()

logllk_div_time_one *Compute loglikelihood of divergence times for a(t) = c/(1-t)*

Description

Compute loglikelihood of divergence times for a(t) = c/(1-t)

Usage

```
logllk_div_time_one(c, l, r, t)
```

Arguments

c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
l	number of data points to the left
r	number of data points to the right
t	a number in the interval (0, 1) indicating the divergence time

See Also

Other likelihood functions: logllk_ddt_lcm(), logllk_ddt(), logllk_div_time_two(), logllk_lcm(), logllk_location(), logllk_tree_topology()

logllk_div_time_two *Compute loglikelihood of divergence times for a(t) = c/(1-t)^2*

Description

Compute loglikelihood of divergence times for a(t) = c/(1-t)^2

Usage

```
logllk_div_time_two(c, l, r, t)
```

Arguments

c	a positive number for the divergence hyperparameter. A larger value implies earlier divergence on the tree
l	number of data points to the left
r	number of data points to the right
t	a number in the interval (0, 1) indicating the divergence time

See Also

Other likelihood functions: logllk_ddt_lcm(), logllk_ddt(), logllk_div_time_one(), logllk_lcm(), logllk_location(), logllk_tree_topology()

`logllk_lcm` *Calculate loglikelihood of the latent class model, conditional on tree structure*

Description

Calculate loglikelihood of the latent class model, conditional on tree structure

Usage

```
logllk_lcm(
  response_matrix,
  leaf_data,
  prior_class_probability,
  prior_dirichlet,
  ClassItem,
  Class_count
)
```

Arguments

`response_matrix`	a N by J binary matrix, where the i,j-th element is the response of item j for individual i
`leaf_data`	a K by J matrix of logit(theta_kj)
`prior_class_probability`	a length K vector, where the k-th element is the probability of assigning an individual to class k. It does not have to sum up to 1
`prior_dirichlet`	a vector of length K. The Dirichlet prior of class probabilities
`ClassItem`	a K by J matrix, where the k,j-th element counts the number of individuals that belong to class k have a positive response to item j
`Class_count`	a length K vector, where the k-th element counts the number of individuals belonging to class k

Value

a numeric of loglikelihood

See Also

Other likelihood functions: `logllk_ddt_lcm()`, `logllk_ddt()`, `logllk_div_time_one()`, `logllk_div_time_two()`, `logllk_location()`, `logllk_tree_topology()`

`logllk_location` *Compute log likelihood of parameters*

Description

Compute the marginal log likelihood of the parameters on the leaves of a tree

Usage

```
logllk_location(
  tree_phylo4d,
  Sigma_by_group,
  item_membership_list,
  dist_mat = NULL,
  tol = 1e-07
)
```

Arguments

`tree_phylo4d`	a "phylo4d" object
`Sigma_by_group`	a vector of diffusion variances of G groups
`item_membership_list`	a list of G elements, where the g-th element contains the column indices of `data` corresponding to items in major group g
`dist_mat`	a tree-structured covariance matrix from a given tree. Default is NULL. If given a matrix, then computation of the covariance matrix will be skipped to save time. This is useful in the Metropolis-Hasting algorithm when the previous proposal is not accepted.
`tol`	a small number to prevent underflow when computing eigenvalues

Value

A list of two elements: a numeric loglikelihood, a covariance matrix of the input tree

See Also

Other likelihood functions: `logllk_ddt_lcm()`, `logllk_ddt()`, `logllk_div_time_one()`, `logllk_div_time_two()`, `logllk_lcm()`, `logllk_tree_topology()`

logllk_tree_topology *Compute loglikelihood of the tree topology*

Description

Compute loglikelihood of the tree topology

Usage

```
logllk_tree_topology(l, r)
```

Arguments

l	number of data points to the left
r	number of data points to the right

See Also

Other likelihood functions: logllk_ddt_lcm(), logllk_ddt(), logllk_div_time_one(), logllk_div_time_two(), logllk_lcm(), logllk_location()

log_expit *Numerically accurately compute f(x) = log(x / (1/x)).*

Description

Numerically accurately compute f(x) = log(x / (1/x)).

Usage

```
log_expit(x)
```

Arguments

x	a value or a numeric vector between 0 and 1 (exclusive)

Value

a number or rea-valued vector

`plot.summary.ddt_lcm` *Plot the MAP tree and class profiles of summarized DDT-LCM results*

Description

Plot the MAP tree and class profiles of summarized DDT-LCM results

Usage

```
## S3 method for class 'summary.ddt_lcm'
plot(
  x,
  log = TRUE,
  plot_option = c("all", "profile", "tree"),
  item_name_list = NULL,
 color_palette = c("#E69F00", "#56B4E9", "#009E73", "#000000", "#0072B2", "#D55E00",
    "#CC79A7", "#F0E442", "#999999"),
  ...
)
```

Arguments

`x`	a "summary.ddt_lcm" object
`log`	Default argument passed to plot(). Not used.
`plot_option`	option to select which part of the plot to return. If "all", return the plot of MAP tree on the left and the plot of class profiles on the right. If "profile", only return the plot of class profiles. If "tree", only return the plot of MAP tree.
`item_name_list`	a named list of G elements, where the g-th element contains a vector of item names for items in `item_membership_list[[g]]`. The name of the g-th element is the name of the major item group.
`color_palette`	a vector of color names. Default is a color-blinded friendly palette.
`...`	Further arguments passed to each method

Value

a ggplot2 object. If plot_option is "all", then a plot with maximum a posterior tree structure on the left and a bar plot of item response probabilities (with 95% credible intervals and class probabilities) on the right. If plot_option is "profile", then only a bar plot of item response probabilities. If plot_option is "tree", then only a plot of the tree structure.

Examples

```
data(result_hchs)
burnin <- 50
summarized_result <- summary(result_hchs, burnin, relabel = TRUE, be_quiet = TRUE)
plot(x = summarized_result, item_name_list = NULL, plot_option = "all")
```

`plot_tree_with_barplot` *Plot the MAP tree and class profiles (bar plot) of summarized DDT-LCM results*

Description

Plot the MAP tree and class profiles (bar plot) of summarized DDT-LCM results

Usage

```
plot_tree_with_barplot(
  tree_with_parameter,
  response_prob,
  item_membership_list,
  item_name_list = NULL,
  class_probability = NULL,
  class_probability_lower = NULL,
  class_probability_higher = NULL,
 color_palette = c("#E69F00", "#56B4E9", "#009E73", "#000000", "#0072B2", "#D55E00",
    "#CC79A7", "#F0E442", "#999999"),
  return_separate_plots = FALSE
)
```

Arguments

`tree_with_parameter`
: a "phylo4d" tree with node parameters

`response_prob`
: a K by J matrix, where the k,j-th element is the response probability of item j for individuals in class k

`item_membership_list`
: a list of G elements, where the g-th element contains the column indices of the observed data matrix corresponding to items in major group g

`item_name_list`
: a named list of G elements, where the g-th element contains a vector of item names for items in `item_membership_list[[g]]`. The name of the g-th element is the name of the major item group.

`class_probability`
: a length K vector, where the k-th element is the probability of assigning an individual to class k. It does not have to sum up to 1

`class_probability_lower`
: a length K vector, 2.5% quantile of posterior the distribution.

`class_probability_higher`
: a length K vector, 97.5% quantile of posterior the distribution.

`color_palette`
: a vector of color names. Default is a color-blinded friendly palette.

return_separate_plots
: If FALSE (default), print the combined plot of MAP tree and class profiles. If TRUE, return the tree plot, class profile plot, and data.table used to create the plots in a list, without printing the combined plot.

Value

a ggplot2 object. A bar plot of item response probabilities.

Examples

```
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_synthetic)
# extract elements into the global environment
list2env(setNames(data_synthetic, names(data_synthetic)), envir = globalenv())
plot_tree_with_barplot(tree_with_parameter, response_prob, item_membership_list)
```

---

plot_tree_with_heatmap

Plot the MAP tree and class profiles (heatmap) of summarized DDT-LCM results

---

Description

Plot the MAP tree and class profiles (heatmap) of summarized DDT-LCM results

Usage

```
plot_tree_with_heatmap(
  tree_with_parameter,
  response_prob,
  item_membership_list
)
```

Arguments

tree_with_parameter
: a "phylo4d" tree with node parameters

response_prob
: a K by J matrix, where the k,j-th element is the response probability of item j for individuals in class k

item_membership_list
: a list of G elements, where the g-th element contains the column indices of the observed data matrix corresponding to items in major group g

Value

a ggplot2 object. A plot with the tree structure on the left and a heatmap of item response probabilities on the right, with indication of item group memberships beneath the heatmap.

Examples

```
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_synthetic)
# extract elements into the global environment
list2env(setNames(data_synthetic, names(data_synthetic)), envir = globalenv())
plot_tree_with_heatmap(tree_with_parameter, response_prob, item_membership_list)
```

`predict.ddt_lcm` *Prediction of class memberships from posterior predictive distributions*

Description

Predict individual class memberships based on posterior predictive distributions. For each posterior sample, let the class memberships be modal assignments. Then aggregate over all posterior samples to obtain the most likely assigned classes.

Usage

```
## S3 method for class 'ddt_lcm'
predict(object, data, burnin = 3000, ...)
```

Arguments

`object`	a ddt_lcm object
`data`	an NxJ matrix of multivariate binary responses, where N is the number of individuals, and J is the number of granular items
`burnin`	number of samples to discard from the posterior chain as burn-ins. Default is 3000.
`...`	Further arguments passed to each method

Value

a list of the following named elements:

`class_assignments` an integer vector of individual predicted class memberships taking values in 1, ..., K

`predictive_probs` a N x K matrix of probabilities, where the (i,k)-th element is the probability that the i-th individual is predicted to belong to class k.

Examples

```
data(result_hchs)
burnin <- 50
predicted <- predict(result_hchs, result_hchs$data, burnin)
```

predict.summary.ddt_lcm *Prediction of class memberships from posterior summaries*

Description

Predict individual class memberships based on posterior summary (point estimates of model parameters). The predicted class memberships are modal assignments.

Usage

```
## S3 method for class 'summary.ddt_lcm'
predict(object, data, ...)
```

Arguments

object	a "summary.ddt_lcm" object
data	an NxJ matrix of multivariate binary responses, where N is the number of individuals, and J is the number of granular items
...	Further arguments passed to each method

Value

a list of the following named elements:

class_assignments an integer vector of individual predicted class memberships taking values in 1, ..., K

predictive_probs a N x K matrix of probabilities, where the (i,k)-th element is the probability that the i-th individual is predicted to belong to class k.

Examples

```
data(result_hchs)
burnin <- 50
summarized_result <- summary(result_hchs, burnin, relabel = TRUE, be_quiet = TRUE)
predicted <- predict(summarized_result, result_hchs$data)
```

print.ddt_lcm *Print out setup of a ddt_lcm model*

Description

Print out setup of a ddt_lcm model

Usage

```
## S3 method for class 'ddt_lcm'
print(x, ...)
```

Arguments

x	a "ddt_lcm" object
...	Further arguments passed to each method

See Also

Other ddt_lcm results: `print.summary.ddt_lcm()`, `summary.ddt_lcm()`

Examples

```
data(result_hchs)
print(result_hchs)
```

print.summary.ddt_lcm *Print out summary of a ddt_lcm model*

Description

Print out summary of a ddt_lcm model

Usage

```
## S3 method for class 'summary.ddt_lcm'
print(x, digits = 3L, ...)
```

Arguments

x	a "summary.ddt_lcm" object
digits	integer indicating the number of decimal places (round) to be used.
...	Further arguments passed to each method

See Also

Other ddt_lcm results: `print.ddt_lcm()`, `summary.ddt_lcm()`

Examples

```
data(result_hchs)
burnin <- 50
summarized_result <- summary(result_hchs, burnin, relabel = TRUE, be_quiet = TRUE)
print(summarized_result)
```

`proposal_log_prob` *Calculate proposal likelihood*

Description

Given an old tree, propose a new tree and calculate the original and proposal tree likelihood in the DDT process

Usage

```
proposal_log_prob(
  old_tree_phylo4,
  tree_kept,
  old_detach_pa_div_time,
  old_pa_detach_node_label,
  old_detach_node_label,
  new_div_time,
  new_attach_root,
  new_attach_to,
  c,
  c_order = 1
)
```

Arguments

`old_tree_phylo4`	the old "phylo4" object
`tree_kept`	the remaining "phylo" tree after detachment
`old_detach_pa_div_time`	a number in (0, 1) indicating the divergence time of the detached node on the old tree
`old_pa_detach_node_label`	a character label of the parent of the detached node on the old tree
`old_detach_node_label`	a character label of the detached node on the old tree
`new_div_time`	a number in (0, 1) indicating the divergence time at which the detached subtree will be re-attached on the proposal tree

`new_attach_root, new_attach_to`
: a character label of the starting and ending nodes of the branch on the proposal tree, which the detached subtree will be re-attached to

`c`
: hyparameter of divergence function a(t)

`c_order`
: equals 1 (default) or 2 to choose divergence function

Value

a list containing the following elements:

`q_new` a "phylo" tree detached from the input tree

`q_old` the remaining "phylo" tree after detachment

`quiet` *Suppress print from cat()*

Description

Suppress print from cat()

Usage

```
quiet(x, be_quiet = TRUE)
```

Arguments

`x`
: evaluation of a statement that may explicitly or implicitly involve cat()

`be_quiet`
: logical. TRUE to suppress print from cat(); FALSE to continue printing

---

`random_detach_subtree` *Metropolis-Hasting algorithm for sampling tree topology and branch lengths from the DDT branching process.*

Description

Randomly detach a subtree from a given tree

Usage

```
random_detach_subtree(tree_phylo4)
```

Arguments

`tree_phylo4`
: a "phylo4" object

Value

a list containing the following elements:

`tree_detached` a "phylo" tree detached from the input tree

`tree_kept` the remaining "phylo" tree after detachment

`pa_detach_node_label` a character label of the parent of the node from which the detachment happens

`pa_div_time` a number in (0, 1) indicating the divergence time of the parent of the detached node

`detach_div_time` a number in (0, 1) indicating the divergence time of the detached node

`detach_node_label` a character label of the parent of the detached node

See Also

Other sample trees: `attach_subtree()`, `reattach_point()`

Examples

```
library(phylobase)
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_synthetic)
# extract elements into the global environment
list2env(setNames(data_synthetic, names(data_synthetic)), envir = globalenv())
detachment <- random_detach_subtree(extractTree(tree_with_parameter))
```

---

reattach_point *Attach a subtree to a given DDT at a randomly selected location*

Description

Attach a subtree to a given DDT at a randomly selected location

Usage

```
reattach_point(tree_kept, c, c_order = 1, theta = 0, alpha = 0)
```

Arguments

`tree_kept`	the tree to be attached to
`c`	hyparameter of divergence function a(t)
`c_order`	equals 1 (default) or 2 to choose divergence function a(t) = c/(1-t) or c/(1-t)^2.
`alpha, theta`	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) For DDT, alpha = theta = 0. For general multifurcating tree from a Pitman-Yor process, specify positive values to alpha and theta. It is, however, recommended using alpha = theta = 0 in inference because multifurcating trees have not been tested rigorously.

Value

a list of the following objects:

`div_time` a numeric value of newly sampled divergence time. Between 0 and 1.

`root_node` a character. Label of the root node of `tree_kept`.

`root_child` a character. Label of the child node of the root of `tree_kept`.

`div_dist_to_root_child` a N-vector with integer entries from 1, ..., K. The initial values for individual class assignments.

See Also

Other sample trees: `attach_subtree()`, `random_detach_subtree()`

---

result_hchs *Result of fitting DDT-LCM to a semi-synthetic data example*

Description

This is a "ddtlcm" object obtained from running `ddtlcm_fit` to a semi-synthetic dataset with 100 posterior samples (for the sake of time). See `ddtlcm_fit` for description of the object.

Usage

```
data(result_hchs)
```

Format

A list with 8 elements

---

sample_class_assignment *Sample individual class assignments Z_i, i = 1, ..., N*

Description

Sample individual class assignments Z_i, i = 1, ..., N

Usage

```
sample_class_assignment(
  data,
  leaf_data,
  a_pg,
  auxiliary_mat,
  class_probability
)
```

Arguments

data	a N by J binary matrix, where the i,j-th element is the response of item j for individual i
leaf_data	a K by J matrix of logit(theta_kj)
a_pg	a N by J matrix of hyperparameters of the generalized logistic distribution
auxiliary_mat	a N by J matrix of truncated normal variables from previous iteration
class_probability	a length K vector, where the k-th element is the probability of assigning an individual to class k. It does not have to sum up to 1

Value

a vector of length N, where the i-th element is the class assignment of individual i

---

sample_c_one *Sample divergence function parameter c for a(t) = c / (1-t) through Gibbs sampler*

Description

Sample divergence function parameter c for a(t) = c / (1-t) through Gibbs sampler

Usage

```
sample_c_one(shape0, rate0, tree_structure)
```

Arguments

shape0	shape of the Inverse-Gamma prior
rate0	rate of the Inverse-Gamma prior
tree_structure	a data.frame containing the divergence times and number of data points to the left and right branches of internal nodes on the tree

Value

a numeric value of the newly sampled c

`sample_c_two` *Sample divergence function parameter c for a(t) = c / (1-t)^2 through Gibbs sampler*

Description

Sample divergence function parameter c for a(t) = c / (1-t)^2 through Gibbs sampler

Usage

```
sample_c_two(shape0, rate0, tree_structure)
```

Arguments

`shape0`	shape of the Inverse-Gamma prior
`rate0`	rate of the Inverse-Gamma prior
`tree_structure`	a data.frame containing the divergence times and number of data points to the left and right branches of internal nodes on the tree

Value

a numeric value of the newly sampled c

`sample_leaf_locations_pg` *Sample the leaf locations and Polya-Gamma auxilliary variables*

Description

Sample the leaf locations and Polya-Gamma auxilliary variables

Usage

```
sample_leaf_locations_pg(
  item_membership_list,
  dist_mat_old,
  Sigma_by_group,
  pg_mat,
  a_pg,
  auxiliary_mat,
  auxiliary_mat_range,
  class_assignments
)
```

Arguments

item_membership_list	a vector of G elements, each indicating the number of items in this group
dist_mat_old	a list of leaf covariance matrix from the previous iteration. The list has length G, the number of item groups
Sigma_by_group	a vector of length G, each denoting the variance of the brownian motion
pg_mat	a K by J matrix of PG variables from the previous iteration
a_pg	a N by J matrix of hyperparameters of the generalized logistic distribution
auxiliary_mat	a N by J matrix of truncated normal variables from previous iteration
auxiliary_mat_range	a list of two named elements: lb and ub. Each is an N by J matrix of the lower/upper bounds of the truncated normal variables.
class_assignments	an integer vector of length N for the individual class assignments. Each element takes value in 1, ..., K.

Value

a named list of three matrices: the newly sampled leaf parameters, the Polya-gamma random variables, and the auxiliary truncated normal variables

---

sample_sigmasq *Sample item group-specific variances through Gibbs sampler*

Description

Sample item group-specific variances through Gibbs sampler

Usage

```
sample_sigmasq(shape0, rate0, dist_mat, item_membership_list, locations)
```

Arguments

shape0	a vector of G elements, each being the shape of the Inverse-Gamma prior of group g
rate0	a vector of G elements, each being the rate of the Inverse-Gamma prior of group g
dist_mat	a list, containing the KxK tree-structured matrix of leaf nodes, where K is the number of leaves / latent classes, and SVD components
item_membership_list	a vector of G elements, each indicating the number of items in this group
locations	a KxJ matrix of leaf parameters

Value

a numeric vector of G elements, each being the newly sampled variance of the latent location of this group

---

`sample_tree_topology` *Sample a new tree topology using Metropolis-Hastings through randomly detaching and re-attaching subtrees*

Description

Sample a new tree topology using Metropolis-Hastings through randomly detaching and re-attaching subtrees

Usage

```
sample_tree_topology(
  tree_phylo4d_old,
  Sigma_by_group,
  item_membership_list,
  c,
  c_order = 1,
  tree_structure_old = NULL,
  dist_mat_old = NULL
)
```

Arguments

`tree_phylo4d_old`	a phylo4d object of tree from the previous iteration
`Sigma_by_group`	a vector of diffusion variances of G groups from the previous iteration
`item_membership_list`	a vector of G elements, each indicating the number of items in this group
`c`	hyparameter of divergence function a(t)
`c_order`	equals 1 (default) or 2 to choose divergence function
`tree_structure_old`	a data.frame of tree structure from the previous iteration. Each row contains information of an internal node, including divergence times, number of data points traveling through the left and right branches
`dist_mat_old`	a list of leaf covariance matrix from the previous iteration. The list has length G, the number of item groups

Value

a numeric vector of G elements, each being the newly sampled variance of the latent location of this group

simulate_DDT_tree *Simulate a tree from a DDT process. Only the tree topology and branch lengths are simulated, without node parameters.*

Description

Simulate a tree from a DDT process. Only the tree topology and branch lengths are simulated, without node parameters.

Usage

```
simulate_DDT_tree(K, c, c_order = 1, alpha = 0, theta = 0)
```

Arguments

K	number of leaves (classes) on the tree
c	hyparameter of divergence function a(t)
c_order	equals 1 (default) or 2 to choose divergence function a(t) = c/(1-t) or c/(1-t)^2.
alpha, theta	hyparameter of branching probability a(t) Gamma(m-alpha) / Gamma(m+1+theta) For DDT, alpha = theta = 0. For general multifurcating tree from a Pitman-Yor process, specify positive values to alpha and theta. It is, however, recommended using alpha = theta = 0 in inference because multifurcating trees have not been tested rigorously.

Value

A class "phylo" tree with K leaves. The leaf nodes are labeled "v1", ..., "vK", root node "u1", and internal nodes "u2", ..., "uK". Note that this tree does not contain any node parameters.

References

Knowles, D. A., & Ghahramani, Z. (2014). Pitman yor diffusion trees for bayesian hierarchical clustering. IEEE transactions on pattern analysis and machine intelligence, 37(2), 271-289.

See Also

Other simulate DDT-LCM data: simulate_lcm_given_tree(), simulate_lcm_response(), simulate_parameter_on_tr

Examples

```
K <- 6
c <- 5
c_order <- 1
tree1 <- simulate_DDT_tree(K, c, c_order)
tree2 <- simulate_DDT_tree(K, c, c_order, alpha = 0.4, theta = 0.1)
tree3 <- simulate_DDT_tree(K, c, c_order, alpha = 0.8, theta = 0.1)
```

`simulate_lcm_given_tree` *Simulate multivariate binary responses from a latent class model given a tree*

Description

Generate multivariate binary responses from the following process: For individual i = 1, ..., N, Z_i ~ Categorical_K(prior_class_probability) For item j = 1, ..., J, Y_ij | Z_i = k ~ Binomial(class_item_probability_kj)

Usage

```
simulate_lcm_given_tree(
  tree_phylo,
  N,
  class_probability = 1,
  item_membership_list,
  Sigma_by_group = NULL,
  root_node_location = 0,
  seed_parameter = 1,
  seed_response = 1
)
```

Arguments

`tree_phylo`	a "phylo" tree with K leaves
`N`	number of individuals
`class_probability`	a length K vector, where the k-th element is the probability of assigning an individual to class k. It does not have to sum up to 1
`item_membership_list`	a list of G elements, where the g-th element contains the column indices of the observed data matrix corresponding to items in major group g
`Sigma_by_group`	a length-G vector for the posterior mean group-specific diffusion variances.
`root_node_location`	the coordinate of the root node parameter. By default, the node parameter initiates at the origin so takes value 0. If a value, then the value will be repeated into a length J vector. If a vector, it must be of length J.
`seed_parameter`	an integer random seed to generate parameters given the tree
`seed_response`	an integer random seed to generate multivariate binary observations from LCM

Value

a named list of the following elements:

`tree_with_parameter` a "phylo4d" tree with K leaves.

`response_prob` a K by J matrix, where the k,j-th element is the response probability of item j for individuals in class k

`response_matrix` a K by J matrix with entries between `0` and 1 for the item response probabilities.

`class_probability` a K-vector with entries between 0 and 1 for the class probabilities. Entries should be nonzero and sum up to 1, or otherwise will be normalized

`class_assignments` a N-vector with integer entries from 1, ..., K. The initial values for individual class assignments.

`Sigma_by_group` a G-vector greater than 0. The initial values for the group-specific diffusion variances.

`c` a value greater than 0. The initial values for the group-specific diffusion variances.

`item_membership_list` same as input

See Also

Other simulate DDT-LCM data: `simulate_DDT_tree()`, `simulate_lcm_response()`, `simulate_parameter_on_tree()`

Examples

```
# load the MAP tree structure obtained from the real HCHS/SOL data
data(data_hchs)
# unlist the elements into variables in the global environment
list2env(setNames(data_hchs, names(data_hchs)), envir = globalenv())
# number of individuals
N <- 496
# set random seed to generate node parameters given the tree
seed_parameter = 1
# set random seed to generate multivariate binary observations from LCM
seed_response = 1
# simulate data given the parameters
sim_data <- simulate_lcm_given_tree(tree_phylo, N,
                                    class_probability, item_membership_list, Sigma_by_group,
                                root_node_location = 0, seed_parameter = 1, seed_response = 1)
```

---

simulate_lcm_response *Simulate multivariate binary responses from a latent class model*

Description

Generate multivariate binary responses from the following process: For individual i = 1, ..., N, Z_i ~ Categorical_K(prior_class_probability) For item j = 1, ..., J, Y_ij | Z_i = k ~ Binomial(class_item_probability_kj)

Usage

```
simulate_lcm_response(N, response_prob, class_probability)
```

Arguments

N	number of individuals
response_prob	a K by J matrix, where the k,j-th element is the response probability of item j for individuals in class k
class_probability	a length K vector, where the k-th element is the probability of assigning an individual to class k. It does not have to sum up to 1

Value

a named list of the following elements:

response_matrix a K by J matrix with entries between 0 and 1 for the item response probabilities.

class_probability a K-vector with entries between 0 and 1 for the class probabilities. Entries should be nonzero and sum up to 1, or otherwise will be normalized

See Also

Other simulate DDT-LCM data: simulate_DDT_tree(), simulate_lcm_given_tree(), simulate_parameter_on_tree()

Examples

```
# number of latent classes
K <- 6
# number of items
J <- 78
response_prob <- matrix(runif(K*J), nrow = K)
class_probability <- rep(1/K, K)
# number of individuals
N <- 100
response_matrix <- simulate_lcm_response(N, response_prob, class_probability)
```

---

simulate_parameter_on_tree

Simulate node parameters along a given tree.

Description

Simulate node parameters along a given tree.

Usage

```
simulate_parameter_on_tree(
  tree_phylo,
  Sigma_by_group,
  item_membership_list,
  root_node_location = 0
)
```

Arguments

tree_phylo	a "phylo" object containing the tree topology and branch lengths
Sigma_by_group	a G-vector greater than 0. The initial values for the group-specific diffusion variances
item_membership_list	a list of G elements, where the g-th element contains the indices of items in major group g
root_node_location	the coordinate of the root node parameter. By default, the node parameter initiates at the origin so takes value 0. If a value, then the value will be repeated into a length J vector. If a vector, it must be of length J.

Value

A class "phylo4d" tree with K leaves with node parameters. The leaf nodes are labeled "v1", ..., "vK", root node "u1", and internal nodes "u2", ..., "uK".

See Also

Other simulate DDT-LCM data: simulate_DDT_tree(), simulate_lcm_given_tree(), simulate_lcm_response()

Examples

```
library(ape)
tr_txt <- "(((v1:0.25, v2:0.25):0.65, v3:0.9):0.1);"
tree <- read.tree(text = tr_txt)
tree$node.label <- paste0("u", 1:Nnode(tree))
plot(tree, show.node.label = TRUE)
# create a list of item membership indices of 7 major groups
item_membership_list <- list()
num_items_per_group <- c(rep(10, 5), 15, 15)
G <- length(num_items_per_group)
j <- 0
for (g in 1:G) {
  item_membership_list[[g]] <- (j+1):(j+num_items_per_group[g])
  j <- j+num_items_per_group[g]
}
# variance of logit response probabilities of items in each group
Sigma_by_group <- c(rep(0.6**2, 5), rep(2**2, 2)) #rep(1**2, G)
set.seed(50)
tree_with_parameter <- simulate_parameter_on_tree(tree, Sigma_by_group, item_membership_list)
```

---

summary.ddt_lcm *Summarize the output of a ddt_lcm model*

Description

Summarize the output of a ddt_lcm model

Usage

```
## S3 method for class 'ddt_lcm'
summary(object, burnin = 3000, relabel = TRUE, be_quiet = FALSE, ...)
```

Arguments

object	a "ddt_lcm" object
burnin	number of samples to discard from the posterior chain as burn-ins. Default is 3000.
relabel	If TRUE, perform post-hoc label switching using the Equivalence Classes Representatives (ECR) method to solve non-identifiability issue in mixture models. If FALSE, no label switching algorithm will be performed.
be_quiet	If TRUE, do not print information during summarization. If FALSE, print label switching information and model summary.
...	Further arguments passed to each method

Value

an object of class "ddt_lcm"; a list containing the following elements:

`tree_map` the MAP tree of "phylo4d" class

`tree_Sigma` the tree-structured covariance matrix associated with `tree_map`

`response_probs_summary`, `class_probs_summary`, `Sigma_summary`, `c_summary` each is a matrix with 7 columns of summary statistics of posterior chains, including means, standard deviation, and five quantiles. In particular, for the summary of item response probabilities, each row name theta_k,g,j represents the response probability of a person in class k to consume item j in group g

`max_llk_full` a numeric value of the maximum log-likelihood of the full model (tree and LCM)

`max_llk_lcm` a numeric value of the maximum log-likelihood of the LCM only

`Z_samples` a `N` x `total_iters` integer matrix of posterior samples of individual class assignments

`Sigma_by_group_samples` a `G` x `total_iters` matrix of posterior samples of diffusion variances

`c_samples` a `total_iters` vector of posterior samples of divergence function hyperparameter

`loglikelihood` a `total_iters` vector of log-likelihoods of the full model

`loglikelihood_lcm` a `total_iters` vector of log-likelihoods of the LCM model only

`setting` a list of model setup information. See `ddtlcm_fit`

`controls` a list of model controls. See `ddtlcm_fit`

`data` the input data matrix

See Also

Other ddt_lcm results: `print.ddt_lcm()`, `print.summary.ddt_lcm()`

Examples

```
# load the result of fitting semi-synthetic data with 100 (for the sake of time) posterior samples
data(result_hchs)
summarized_result <- summary(result_hchs, burnin = 50, relabel = TRUE, be_quiet = TRUE)
```

WAIC *Compute WAIC*

Description

Compute the Widely Applicable Information Criterion (WAIC), also known as the Widely Available Information Criterion or the Watanable-Akaike, of Watanabe (2010).

Usage

```
WAIC(llk_matrix)
```

Arguments

llk_matrix	a N x S matrix, where N is the number of individuals and S is the number of posterior samples

Value

a named list

Index