

Package: thamesmix (via r-universe)

June 5, 2026

Type Package

Title Truncated Harmonic Mean Estimator of the Marginal Likelihood for Mixtures

Version 0.1.3

Description Implements the truncated harmonic mean estimator (THAMES) of the reciprocal marginal likelihood for uni- and multivariate mixture models using posterior samples and unnormalized log posterior values via reciprocal importance sampling. Metodiev, Irons, Perrot-Dockès, Latouche & Raftery (2025) [doi:10.48550/arXiv.2504.21812](https://doi.org/10.48550/arXiv.2504.21812).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports stats, sparsediscrim, quadprog, igraph, gor, Rfast, mvtnorm, combinat, withr

VignetteBuilder knitr

Suggests multimode, knitr, bayesmix, label.switching, LaplacesDemon, markdown

Config/pak/sysreqs libgmp-dev make libxml2-dev

Repository <https://m-metodiev.r-universe.dev>

Date/Publication 2025-10-08 12:15:10 UTC

RemoteUrl <https://github.com/m-metodiev/thamesmix>

RemoteRef HEAD

RemoteSha d888561abc82a2e0687fcff8d13cab139f763ae8

Contents

alltopsorts_recursion	2
compute_nobile_identity	2
overlapgraph	3
thames_mixtures	5

Index**9**

alltopsorts_recursion *all topological orderings of a DAG*

Description

This function computes all topological orderings of a graph using the recursive algorithm described in Knuth and Szwarcfiter (1974).

Usage

```
alltopsorts_recursion(n, adj_list)
```

Arguments

n	number of nodes in the DAG
adj_list	edges given as an adjacency list

Value

Returns a list of topological orderings.

References

Knuth, D. E. and J. L. Szwarcfiter (1974). A structured program to generate all topological sorting arrangements. Information Processing Letters 2(6), 153–157.

Examples

```
n = 4
alltopsorts_recursion(n, list(c(1,3),c(2,4)))
```

compute_nobile_identity

Nobile's identity for the marginal likelihood

Description

This function uses the identity from Nobile (2004, 2007) to compute an estimate of the marginal likelihood for a mixture model with G components given an estimate of the marginal likelihood for a mixture model with G-1 components and an estimate of the proportion of empty components.

Usage

```
compute_nobile_identity(logZhatGminus1, p0hat_value, G, dirichlet_vec, n)
```

Arguments

logZhatGminus1	estimate of the marginal likelihood for G-1
p0hat_value	estimate of the proportion of empty components
G	number of components
dirichlet_vec	hyperparameter-vector of the dirichlet prior
n	size of the data

Value

estimate of the marginal likelihood for G

References

Nobile, A. (2004). On the posterior distribution of the number of components in a finite mixture. *The Annals of Statistics* 32(5), 2044–2073.

Nobile, A. (2007). Bayesian finite mixtures: a note on prior specification and posterior computation. arXiv preprint arXiv:0711.0458.

Martin Metodiev, Nicholas J. Irons, Marie Perrot-Dockès, Pierre Latouche, Adrian E. Raftery. "Easily Computed Marginal Likelihoods for Multivariate Mixture Models Using the THAMES Estimator." arXiv preprint arXiv:2504.21812.

Examples

```
# computes log marginal likelihood of the Swiss banknote dataset
# for G=4, given the settings in Metodiev et al. (2025)
compute_nobile_identity(logZhatGminus1 = -909.49,
p0hat_value = 1/4,
dirichlet_vec = rep(1,4),
n=200)
```

overlapgraph

Estimator of the overlap graph

Description

This function computes the overlap graph for mixture models.

Usage

```
overlapgraph(sims)
```

Arguments

sims	n_simul x G x (u+1) array of parameters sampled from the posterior, where n_simul is the number of simulations from the posterior, G is the number of components, u is the number of mixture component parameters (parameter u+1 is the mixture weight)
------	---

Value

Returns a named list with the following elements:

graph, the overlap graph
co, the criterion of overlap

References

Martin Metodiev, Nicholas J. Irons, Marie Perrot-Dockès, Pierre Latouche, Adrian E. Raftery. "Easily Computed Marginal Likelihoods for Multivariate Mixture Models Using the THAMES Estimator." arXiv preprint arXiv:2504.21812.

Examples

```
# toy sample from the posterior
mus = rbind(c(17.67849, 21.46734),
            c(17.67849, 21.46734),
            c(16.98067, 21.11391),
            c(20.58628, 21.22104),
            c(17.38332, 21.37224),
            c(16.43644, 21.19085),
            c(19.49676, 21.28964),
            c(17.82287, 21.22475),
            c(18.06050, 21.36945),
            c(18.70759, 21.60244),
            c(15.93795, 21.04681),
            c(16.23184, 20.96049))
sigmasqus = rbind(c(46.75089, 3.660171),
                  c(58.44208, 3.026577),
                  c(63.19334, 4.090872),
                  c(87.02758, 2.856063),
                  c(82.34268, 3.760550),
                  c(50.92386, 2.380784),
                  c(49.51412, 3.605798),
                  c(38.67681, 3.362407),
                  c(49.59170, 3.130254),
                  c(63.41569, 2.475669),
                  c(65.95225, 3.927501),
                  c(47.22989, 5.465702))
taus = rbind(c(0.2653882, 0.7346118),
             c(0.2560075, 0.7439925),
             c(0.2371868, 0.7628132),
             c(0.2998265, 0.7001735),
             c(0.3518301, 0.6481699),
             c(0.2840316, 0.7159684),
             c(0.2060193, 0.7939807),
             c(0.2859257, 0.7140743),
             c(0.2420695, 0.7579305),
             c(0.2466622, 0.7533378),
             c(0.2726186, 0.7273814),
             c(0.2738916, 0.7261084))
sims = array(dim=c(12,2,3))
```

```

sims[,1] = mus
sims[,2] = sigmasqus
sims[,3] = taus

overlapgraph(sims)$co

```

thames_mixtures	<i>THAMES estimator of the reciprocal log marginal likelihood for mixture models</i>
-----------------	--

Description

This function computes the THAMES estimate of the reciprocal log marginal likelihood for mixture models using posterior samples and unnormalized log posterior values.

Usage

```

thames_mixtures(
  logpost,
  sims,
  n_samples = NULL,
  c_opt = NULL,
  type = "simple",
  seed = NULL,
  lps = NULL,
  lps_unif = NULL,
  max_iters = Inf
)

```

Arguments

logpost	function logpost(sims,G) to compute lps with input "sims"
sims	$n_simul \times G \times (u+1)$ array of parameters sampled from the posterior, where n_simul is the number of simulations from the posterior, G is the number of components, u is the number of mixture component parameters (parameter $u+1$ is the mixture weight)
n_samples	integer, number of posterior samples
c_opt	radius of the ellipsoid used to compute the THAMES
type	THAMES variant ("simple", "permutations", or "standard")
seed	a seed
lps	values of the unnormalized log posterior density
lps_unif	values of the unnormalized log posterior density, evaluated on a uniform sample on the posterior ellipsoid
max_iters	maximum number of shrinkage iterations

Value

Returns a named list with the following elements:

theta_hat, posterior mean
 sigma_hat, posterior covariance matrix
 log_det_sigma_hat, log-determinant of sigma_hat
 logvolA, log-volume of the ellipsoid
 log_zhat_inv, log-reciprocal-marginal likelihood
 log_zhat_inv_L, lower bound
 log_zhat_inv_U, upper bound
 alpha, HPD-region correction
 len_perms, number of permutations evaluated
 log_cor, log-correction of the volume of the ellipsoid
 etas, Monte-Carlo sample on the ellipsoid
 graph, the overlap graph for G
 se, standard_error
 phi, ar(1) model parameter
 c_opt, radius of the ellipsoid
 d_par, dimension of the parameter
 G, number of mixture components
 scaling, list of fit of QDA (means, covariances)
 co, the criterion of overlap

References

Martin Metodiev, Nicholas J. Irons, Marie Perrot-Dockès, Pierre Latouche, Adrian E. Raftery. "Easily Computed Marginal Likelihoods for Multivariate Mixture Models Using the THAMES Estimator." arXiv preprint arXiv:2504.21812.

Examples

```
y = c(9.172, 9.350, 9.483, 9.558, 9.775, 10.227, 10.406, 16.084, 16.170,
      18.419, 18.552, 18.600, 18.927, 19.052, 19.070, 19.330, 19.343, 19.349,
      19.440, 19.473, 19.529, 19.541, 19.547, 19.663, 19.846, 19.856, 19.863,
      19.914, 19.918, 19.973, 19.989, 20.166, 20.175, 20.179, 20.196, 20.215,
      20.221, 20.415, 20.629, 20.795, 20.821, 20.846, 20.875, 20.986, 21.137,
      21.492, 21.701, 21.814, 21.921, 21.960, 22.185, 22.209, 22.242, 22.249,
      22.314, 22.374, 22.495, 22.746, 22.747, 22.888, 22.914, 23.206, 23.241,
      23.263, 23.484, 23.538, 23.542, 23.666, 23.706, 23.711, 24.129, 24.285,
      24.289, 24.366, 24.717, 24.990, 25.633, 26.690, 26.995, 32.065, 32.789,
      34.279)
```

```
R <- diff(range(y))
m <- mean(range(y))
```

```

# likelihood
loglik_gmm <- function(sims,G){
  mus = sims[,1]
  sigma_squs = sims[,2]
  pis = sims[,3]
  log_single_y = Vectorize(function(x)
    log(rowSums(sapply(1:G,
      function(g) pis[,g]*dnorm(x,mus[,g],sqrt(sigma_squs[,g])))))
    )
  )
  res = suppressWarnings(rowSums(log_single_y(y)))
  return(rowSums(log_single_y(y)))
}

# prior
logprior_gmm_marginal <- function(sims,G) {
  mus = sims[,1]
  sigma_squs = sims[,2]
  pis = sims[,3]

  l_mus <- rowSums(sapply(1:G, function(g) dnorm(mus[,g], mean = m, sd = R,
    log = TRUE)))
  l_pis <- LaplacesDemon::ddirichlet(1:G/G, rep(1,G),log=TRUE)
  l_sigma_squs <- lgamma(2*G+0.2) - lgamma(0.2) +
    0.2*log(10/R^2) - (2*G+0.2) * log(rowSums(sigma_squs^(-1))+10/R^2) -
    3*rowSums(log(sigma_squs))
  return(l_mus + l_pis + l_sigma_squs)
}

# unnormalized log-posterior density
logpost = function(sims){
  G = dim(sims)[2]
  mus = sims[,1:G,1]
  # apply exp transform
  sims[,1:G,2] = sims[,1:G,2]
  sigma_squs = sims[,1:G,2]
  pis = sims[,1:G,3]

  # set to 0 outside of support
  if(G>2){
    mask = (((pis > 0) & (rowSums(pis[,1:(G-1)])<=1)) & (sigma_squs>0))
  }else{
    mask = (((pis > 0) & (pis[,1]<=1)) & (sigma_squs>0))
  }
  l_total = suppressWarnings(loglik_gmm(sims,G)+
    logprior_gmm_marginal(sims,G))
  l_total[exp(rowSums(log(mask)))==0] = -Inf
  return(l_total)
}

# toy sample from the posterior
mus = rbind(c(17.67849, 21.46734),
  c(17.67849, 21.46734),

```

```
c(16.98067, 21.11391),
c(20.58628, 21.22104),
c(17.38332, 21.37224),
c(16.43644, 21.19085),
c(19.49676, 21.28964),
c(17.82287, 21.22475),
c(18.06050, 21.36945),
c(18.70759, 21.60244),
c(15.93795, 21.04681),
c(16.23184, 20.96049))
sigmasqus = rbind(c(46.75089, 3.660171),
                  c(58.44208, 3.026577),
                  c(63.19334, 4.090872),
                  c(87.02758, 2.856063),
                  c(82.34268, 3.760550),
                  c(50.92386, 2.380784),
                  c(49.51412, 3.605798),
                  c(38.67681, 3.362407),
                  c(49.59170, 3.130254),
                  c(63.41569, 2.475669),
                  c(65.95225, 3.927501),
                  c(47.22989, 5.465702))
taus = rbind(c(0.2653882, 0.7346118),
             c(0.2560075, 0.7439925),
             c(0.2371868, 0.7628132),
             c(0.2998265, 0.7001735),
             c(0.3518301, 0.6481699),
             c(0.2840316, 0.7159684),
             c(0.2060193, 0.7939807),
             c(0.2859257, 0.7140743),
             c(0.2420695, 0.7579305),
             c(0.2466622, 0.7533378),
             c(0.2726186, 0.7273814),
             c(0.2738916, 0.7261084))
sims = array(dim=c(12,2,3))
sims[, ,1] = mus
sims[, ,2] = sigmasqus
sims[, ,3] = taus

# estimate of the log marginal likelihood
-thames_mixture(logpost,sims)$log_zhat_inv
```

Index

`alltopsorts_recursion`, [2](#)

`compute_nobile_identity`, [2](#)

`overlapgraph`, [3](#)

`thames_mixtures`, [5](#)