Package 'UWHAM'

July 21, 2025

Type Package

Title Unbinned Weighted Histogram Analysis Method (UWHAM)

Version 1.1

Date 2022-05-19

Author Zhiqiang Tan and Emilio Gallicchio

Maintainer Zhiqiang Tan <ztan@stat.rutgers.edu>

URL http://www.stat.rutgers.edu/~ztan,
https://www.compmolbiophysbc.org

Description A method for estimating log-normalizing constants (or free energies) and expectations from multiple distributions (such as multiple generalized ensembles).

Depends R (>= 2.9.1), trust

License GPL (>= 2)

Repository CRAN

Date/Publication 2022-05-20 14:00:02 UTC

NeedsCompilation no

Contents

UWHAM-package — *A R package for the unbinned weighted histogram analysis method (UWHAM)*

Description

UWHAM for estimating log-normalizing constants (or free energies) and expectations from multiple distributions (such as multiple generalized ensembles).

Details

The R package UWHAM – version 1.0 can be used for two main tasks:

- to estimate log-normalizing constants (or free energies) from multiple distributions,
- to estimate expectations from multiple distributions.

There are 3 main functions:

- `uwham()`: to estimate free energies and associated variances,
- `uwham.phi()`: to estimate expectations and associated variances,
- `uwham.boot()`: to estimate variances based on block bootstrap.

The package also provides a function, `histw()`, for plotting weighted histograms.

cyclooctanol — *A simulated tempering dataset of interaction parameters and binding energies*

Description

Time series of interaction parameters and corresponding binding energies from a Hamiltonian hopping simulation of molecular binding.

Usage

```
data(cyclooctanol)
```

Format

A data frame containing 2 columns and 35000 rows.

Details

The dataset contains ligand-receptor interaction parameters (lambda) and binding energies for the equivalent of an Hamiltonian hopping simulation of cyclooctanol binding to beta-cyclodextrin. (Hamiltonian hopping is the same as simulated tempering except that potential parameters are varied rather than temperature.) This illustrative dataset was constructed by concatenating the output of some of the replicas of a Hamiltonian replica exchange simulation of the same complex (Wickstrom et al. 2013). Harmless discontinuities may be observed at the joining points.

The dataset contains 2 columns and 3500 rows. Rows correspond to consecutive time points. The first column `cyclooctanol$V1` contains lambda values (24 discrete values between 0 and 1) and the second column `cyclooctanol$V2` records binding energy values in kcal/mol.

References

Wickstrom, L., Gallicchio, E., He, P., and Levy, R.M. (2013) "Large scale affinity calculations of cyclodextrin host-guest complexes: Understanding the role of reorganization in the molecular recognition process," *Journal of Chemical Theory and Computation*, submitted for publication.

`histw` *Weighted histogram*

Description

This function plots a weighted histogram.

Usage

```
histw(x, w, xaxis, xmin, xmax, ymax,
          bar=TRUE, add=FALSE, col="black", dens=TRUE)
```

Arguments

`x`	A data vector.
`w`	A weight vector, which will be rescaled to sum up to one.
`xaxis`	A vector of cut points.
`xmin`	The minimum of `x` coordinate.
`xmax`	The maximum of `x` coordinate.
`ymax`	The maximum of `y` coordinate.
`bar`	bar plot (if `TRUE`) or line plot.
`add`	if `TRUE`, the plot is added to an existing plot.
`col`	color of lines.
`dens`	if `TRUE`, the histogram has a total area of one.

References

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

Examples

```
# Boltzmann constant
bet <- 1.0/(0.001986209*300.0)

# negative potential function
npot.fcn <- function(x, lam)
   -bet*lam*x

# read data (soft.core)
lam <- c(0.0, 0.001, 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.06, 0.1,
         0.25, 0.5, 0.75, 0.9, 1.0)
m <- length(lam)

data(ligand2.soft)
lig.data <- ligand2.soft$V1

size <- rep(1000, m)
N <- sum(size)

# compute negative potential
neg.pot <- matrix(0, N,m)

for (j in 1:length(lam))
  neg.pot[,j] <- npot.fcn(x=lig.data, lam=lam[j])

# estimate free energies
out <- uwham(logQ=neg.pot, size=size, fisher=TRUE)

-out$ze/bet + 0.71

sqrt(out$ve)/bet

# the bins used to construct weighted histograms
bins <- seq(-35, 400, 2.5)
grid <- c(bins, 2e+3)

W <- out$W /N

par(mfrow=c(1,2))

# Plot the weighted histograms at two lambdas
histw(lig.data, w=W[,8], xaxis=grid, xmin=-35, xmax=200, ymax=.1)

histw(lig.data, w=W[,10], xaxis=grid, xmin=-35, xmax=200, ymax=.1,
      bar=0, add=TRUE, col="red")
```

```
# plot the raw histogram and weighted histogram
histw(lig.data[ (5*1000+1):(6*1000) ], w=rep(1,1000),
      xaxis=grid, xmin=-35, xmax=200, ymax=.04)

histw(lig.data, w=W[,6], xaxis=grid, xmin=-35, xmax=200, ymax=.04,
      bar=0, add=TRUE, col="red")
```

`insert` *Inserting a value into a vector*

Description

This inserts a value `x0` at `d`-th position of `x`.

Usage

```
insert(x, d, x0 = 0)
```

Arguments

`x`	A vector.
`d`	A position in `x`.
`x0`	A value to be inserted.

Value

The resulting vector.

`ligand2.hard` *A dataset of binding energies obtained with a conventional interaction potential*

Description

A dataset for protein-ligand binding with an unmodified energy function.

Usage

```
data(ligand2.hard)
```

Format

A data frame containing 1 column and 18000 rows.

Details

The dataset contains binding energies (in kcal/mol) for the ligand 2/FKBP complex with an unmodified energy function, simulated using replica exchange with 18 lambda values (Tan et al. 2012).

Suppose that the dataset is converted to a `1000` x `18` matrix as follows.

```
data(ligand2.hard)
lig.data <- matrix(ligand2.hard$V1, nrow=1000)
```

The 18 columns of `lig.data` correspond to the raw output of the 18 replicas. A replica is defined as a continuous replica exchange molecular dynamics thread with variable lambda (exchanges are performed by swapping lambda values rather than coordinates). The 1000 rows represent as many sequential time points during the replica exchange trajectory. Each row contains binding energies from each replica taken at the same simulation time (synchronous sampling).

References

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

`ligand2.soft` *A dataset of binding energies with a soft-core interaction potential*

Description

A dataset for protein-ligand binding with a soft energy function.

Usage

```
data(ligand2.soft)
```

Format

A data frame containing 1 column and 15000 rows.

Details

The dataset contains binding energies (in kcal/mol) for the ligand 2/FKBP complex with a soft-core interaction energy function, simulated using replica exchange with 15 lambda values (Tan et al. 2012).

Suppose that the dataset is converted to a `1000` x `15` matrix as follows.

```
data(ligand2.soft)
lig.data <- matrix(ligand2.soft$V1, nrow=1000)
```

The 15 columns of `lig.data` correspond to the 15 lambda states from smallest (lambda=0) to largest (lambda=1). The data were obtained by reordering by lambda the raw output of the molecular dynamics replicas at each observation time. A replica is defined as a continuous replica exchange molecular dynamics thread with variable lambda (exchanges are performed by swapping lambda values rather than coordinates). The 1000 rows represent as many sequential time points during the replica exchange trajectory. Each row contains binding energies from each replica taken at the same simulation time (synchronous sampling).

References

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

`obj.fcn` *The objective function for UWHAM*

Description

This function computes the objective function, its gradient and its hessian matrix for UWHAM.

Usage

```
obj.fcn(ze, logQ, size, base)
```

Arguments

`ze`	A vector of log-normalizing constants (or free energies) for the sampled thermodynamic states.
`logQ`	The matrix of log of unnormalized density ratios for the sampled thermodynamic states over the baseline.
`size`	A vector giving the individual sample sizes for the sampled thermodynamic states.
`base`	The baseline index.

Details

The objective function is convex as discussed in Tan et al. (2012). See also Gill et al. (1998) for related results on biased sampling models.

Value

`value`	The value of the objective function.
`gradient`	The gradient of the objective function.
`hessian`	The hessian matrix of objective function.

References

Gill, R., Vardi, Y., and Wellner, J. (1998) "Large sample theory of empirical distributions in biased sampling models," *Annals of Statistics*, 16, 1069–1112.

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

`uwham` *Unbinned weighted histogram analysis method (UWHAM) for estimating free energies*

Description

This function implements UWHAM for estimating log-normalizing constants (or free energies) and expectations from multiple distributions (such as multiple generalized ensembles) as described in Tan et al. (2012).

Usage

```
uwham(label=NULL, logQ, size=NULL, base = NULL, init = NULL,
      fisher = TRUE)
```

Arguments

`label`	A vector of length `N` of labels between `1` to `M` such that `label[i]=j` when `i`th observation is obtained from `j`th thermodynamic state; either `label` or `size` must be provided; if `fisher=FALSE` and if `label=NULL`, then `label` is set such that the first `size[1]` observations are assumed to be from state `1`, the next `size[2]` observations from state `2`, etc.
`logQ`	`N` x `M` matrix of log unnormalized densities (such as $1/kT$ times negative potential energies), where `N` is the total sample size, i.e., `sum(size)`, and `M` is the number of thermodynamic states for which free energies are to be computed; the `i`th row of `logQ` correspond to `i`th observations and the `j`th column correspond to `j`th thermodynamic state.
`size`	A vector of length `M`, giving the individual sample sizes for the `M` thermodynamic states, ordered as the columns of `logQ`; if `NULL`, then `label` is required and used to compute `size`.
`base`	The baseline index, between `1` to `M`, for the thermodynamic state (with sample size `>0`) whose free energy is set to 0; if `NULL`, then `base` is set to the first index `j` such that `size[j]>0`.
`init`	A vector of length `M`, giving the initial values of the log-normalizing constants (or log of the partition functions); if `NULL`, then `init` is set to the zero vector.
`fisher`	Logical; if `NULL`, no variance estimation; if `TRUE`, variance estimation is based on Fisher information; if `FALSE`, variance estimation is based on the Sandwich variance formula (see the details).

Details

The UWHAM method results from a number of interesting, sometimes independent, developments in physics and statistics. See Kong et al. (2003) for a formal statistical treatment along with earlier references, and Tan et al. (2012) for a more accessible account, presenting the method as a binless extension of the Weighted Histogram Analysis Method (WHAM) widely known in physics and chemistry (e.g., Ferrenberg and Swendsen 1989). The possibility of obtaining free energies from a binless extension of WHAM was noticed by various authors (e.g., Kumar et al. 1992; Newman & Barkema 1999). The binless method was later reintroduced by Shirts & Chodera (2008) to the physics literature and was called the Multi-state Bennet Acceptance Ratio method (MBAR) because it can be interpreted as a multi-state extension of the Bennet Acceptance Ratio (BAR) method. An implementation of MBAR in the Python language developed by Shirts & Chodera is freely available (see https://simtk.org/home/pymbar). The UWHAM package, while adopting an alternative numerical approach, provides identical point estimates compared to the MBAR Python package. In addition, the UWHAM package provides variance estimation based on variance formulas without using generalized inverses or, for correlated data, by block bootstrap.

A typical application of UWHAM involves the computation of the relative free energies of a series of thermodynamic states differing in environmental conditions (temperature for example) and/or Hamiltonian parameters (such as the strength of biasing potentials) from data collected at these thermodynamic states. The method takes as input the reduced energies (such as the inverse temperature times the negative of the potential energy for canonical ensembles differing in temperature) of the observations at all thermodynamic states of interest. In addition to the free energies, the output includes estimates of the thermodynamic weights of the observations for all states to be used for thermodynamic reweighting calculations.

The UWHAM method is statistically optimal in yielding the smallest asymptotic variances, provided that the individual samples are independent and the observations in each sample are also independent (Tan 2004).

To compute point estimates, the method is implemented here by minimizing a convex objective function, as described in Tan et al. (2012). This approach can be more effective than solving the nonlinear equations by the self-consistency or the Newton-Raphson algorithm. Currently, the optimization is done by using the R package *trust*.

Variance estimation provided here is based on the Fisher information or the Sandwich variance formula, as presented in Tan et al. (2012). In contrast with the Sandwich formula, the Fisher information based formula does not require labels indicating which thermodynamic state each observation is obtained from (see the dataset `ligand2.hard`). The analytical variance formulas are consistent when the observations are considered independent. Alternatively, variance estimation can be done by block bootstrap, implemented in `uwham.boot`.

Value

`ze`	The vector of estimated log-normalizing constants (or log of the partition functions).
`ve`	The vector of estimated variances for `ze`, if `fisher!=NULL`.
`Ve`	The estimated variance-covariance matrix for `ze`, if `fisher!=NULL`.
`W`	The `N` x `M` matrix of UWHAM weights for each of the `N` observations at each of the `M` thermodynamic states.

check	The column averages of W[,sampled]; the elements of check should be equal to 1 to indicate a valid convergence of *trust*.
out	The output of *trust* used to minimize the objective function; see help(trust).
label	Same as argument label.
size	Same as argument size.
base	Same as argument base.

References

Ferrenberg, A.M. and Swendsen, R.H. (1989) "Optimized Monte Carlo data analysis," *Physics Review Letters*, 63, 1195-1198.

Kong, A., McCullagh, P., Meng, X.-L., Nicolae, D., and Tan, Z. (2003) "A theory of statistical models for Monte Carlo integration" (with discussion), *Journal of the Royal Statistical Society*, Ser. B, 65, 585-618.

Kumar, S., Bouzida, D., Swendsen, R.H., Kollman, P.A. and Rosenberg, J.M. (1992) "The Weighted Histogram Analysis Method for free-energy calculations on biomolecules. I. The method," *Journal of Computational Chemistry*, 13, 1011-1021.

Newman, M.E.J. and Barkema, G.T. (1999) *Monte Carlo Methods in Statistical Physics*, Oxford University Press, New York.

Shirts, M.R. and J. D. Chodera, J.D. (2008) "Statistically optimal analysis of samples from multiple equilibrium states," *Journal of Chemical Physics*, 129, 124105.

Tan, Z. (2004) "On a likelihood approach for Monte Carlo integration," *Journal of the American Statistical Association*, 99, 1027-1036.

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

Examples

```
########################################
############## example 1 ##############
########################################

# This example illustrates the calculation of the standard free energy
# of binding of a ligand to a protein receptor by means of an alchemical
# perturbation potential of the form lambda*binding_energy(r), where
# lambda is a scaling parameter (lambda=0 corresponds to the
# protein-ligand uncoupled state, and lambda=1 corresponds to the
# coupled state) and binding_energy(r) is the (solvent averaged)
# potential energy of conformation r of the complex relative to one in
# which the receptor and the ligand are rigidly displaced at infinite
# separation. See Gallicchio et al., J. Chem. Theory Comput. 6,
# 2961-2977 (2010), and Tan et al. J. Chem. Phys., 136, 144102 (2012).

# Inverse temperature beta, in kcal/mol^-1
bet <- 1.0/(0.001986209*300.0)
```

```
# negative reduced potential function -beta*lambda*binding_energy.
#  "x" is the binding energy of a structure of the complex and "lam" the
#  value of lambda at which to compute the reduced energy
npot.fcn <- function(x, lam)
   -bet*lam*x

# values of lambda for the calculation with a "hard-core" potential
lam <- c(0.0, 0.000000001, 0.00000001, 0.0000001, 0.000001, 0.00001, 0.0001,
         0.001, 0.01, 0.1, 0.15, 0.25, 0.35, 0.5, 0.6, 0.75, 0.9, 1.0)

#number of alchemical states
m <- length(lam)

# load binding energies
data(ligand2.hard)
lig.data <- ligand2.hard$V1

# 1000 observations at each lambda state
size <- rep(1000, m)

# total sample size
N <- sum(size)

# compute negative potential of each observation at each lambda state
neg.pot <- matrix(0, N,m)
for (j in 1:length(lam))
  neg.pot[,j] <- npot.fcn(x=lig.data, lam=lam[j])

# estimate free energies using UWHAM
out <- uwham(logQ=neg.pot, size=size, fisher=TRUE)

# convergence diagnosis: the elements should be equal to 1
out$check

# the "ze" values are dimensionless free energies, that is the
# log of the partition functions (log Z).
# To obtain thermodynamic free energies, multiply by -kT.
# 0.71 kcal/mol is the standard state correction; see
# Lapelosa et al. J Chem Theory Comput, 8, 44-60 (2012).
-out$ze/bet + 0.71

# variances of free energies from Fisher information
sqrt(out$ve)/bet

# perform block bootstrap for free energies to take into account
# time correlations in the binding energy data
# To save time for package checking, this is not run.
#out.boot <- uwham.boot(proc.type="parallel", block.size=50, boot.size=100,
#                       logQ=neg.pot, size=size)
#
#-out.boot$ze/bet + 0.71
#sqrt(out.boot$ve)/bet
```

```
# estimation of average binding energies and variances
# at lambda = 0.6, 0.75, 0.9, 1.0
state <- 15:18
out.phi <- uwham.phi(phi=lig.data, state=state, out.uwham=out, fisher=TRUE)

out.phi$phi
out.phi$phi.v

# block bootstrap for both free energies and expectations
# To save time for package checking, this is not run.
#out.boot <- uwham.boot(proc.type="parallel", block.size=50, boot.size=100,
#                       logQ=neg.pot, size=size,
#                       phi=lig.data, state=state)
#
#out.boot$phi
#out.boot$phi.v

################################################
## example 2 (unequal and zero sample sizes) ###
################################################

# same calculation as above but with a "soft-core" potential and
# illustrating the ability to compute free energies and expectations
# for states with unequal sample sizes including those with
# zero sample sizes (or states that have not been sampled).
# See Tan et al. J. Chem. Phys., 136, 144102 (2012).

rm(list=ls())

# inverse temperature
bet <- 1.0/(0.001986209*300.0)

# negative potential function
npot.fcn <- function(x, lam)
   -bet*lam*x

# read data (soft core)
lam <- c(0.0, 0.001, 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.06, 0.1,
         0.25, 0.5, 0.75, 0.9, 1.0)
m <- length(lam)

data(ligand2.soft)
lig.data <- ligand2.soft$V1

### unequal and zero sample sizes
size <- c( rep(1000, 5), rep(500, 3), rep(0, 2), rep(1000,5))
subs <- c(rep(TRUE, 5000), rep(c(rep(TRUE,500),rep(FALSE,500)), 3),
          rep(FALSE, 2000), rep(TRUE,5000))
lig.data <- lig.data[subs]

N <- sum(size)

# compute negative potential
```

```
neg.pot <- matrix(0, N,m)
for (j in 1:length(lam))
  neg.pot[,j] <- npot.fcn(x=lig.data, lam=lam[j])

# estimate free energies
out <- uwham(logQ=neg.pot, size=size, fisher=TRUE)

-out$ze/bet + 0.71
sqrt(out$ve)/bet

# block bootstrap for free energies,
# pretending that the data are generated from independent chains.
# To save time for package checking, this is not run.
#out.boot <- uwham.boot(proc.type="indep",
#                       block.size=rep(50,m-2), boot.size=100,
#                       logQ=neg.pot, size=size)
#
#-out.boot$ze/bet + 0.71
#sqrt(out.boot$ve)/bet

########################################
## example 3 (serial tempering data) ##
########################################

rm(list=ls())

# inverse temperature
bet <- 1.0/(0.001986209*300.0)

# negative potential function
npot.fcn <- function(x, lam)
   -bet*lam*x

# lambda states
lam <- c(0.0,0.001,0.002,0.004,0.005,0.006,0.008,0.01,0.02,0.04,
         0.07,0.1,0.25,0.5,0.55,0.6,0.65,0.7,0.75,0.8,
         0.85,0.9,0.95,1.0)
m <- length(lam)

# loads cyclooctanol dataset
data(cyclooctanol)
lig.data <- cyclooctanol$V2

# sample size
N <- length(lig.data)

# state labels based on lambda values
# note that labels=1:m, not 0:(m-1)
state.labels <- factor(cyclooctanol$V1, labels=1:m)

# compute negative potential
neg.pot <- matrix(0, N,m)
for (j in 1:m)
```

```
  neg.pot[,j] <- npot.fcn(x=lig.data, lam=lam[j])

# estimate free energies, note that size=NULL because label is given
out <- uwham(label=state.labels, logQ=neg.pot, fisher=TRUE)

# free energies as a function of lambda, 0.36 kcal/mol is a standard
# state correction
-out$ze/bet + 0.36
sqrt(out$ve)/bet

# block bootstrap for free energies, note that proc.type="serial"
# for simulated tempering data.
# To save time for package checking, this is not run.
#out.boot <- uwham.boot(proc.type="serial", block.size=10, boot.size=100,
#                       label=state.labels, logQ=neg.pot)
#
#-out.boot$ze/bet + 0.36
#sqrt(out.boot$ve)/bet
```

`uwham.boot` *Variance estimation for UWHAM based on block bootstrap*

Description

This function implements variance estimation based on block bootstrap for UWHAM estimates of free energies and expectations.

Usage

```
uwham.boot(proc.type, block.size, boot.size, seed = 0,
           label = NULL, logQ, size=NULL, base = NULL, init = NULL,
           phi = NULL, state = NULL)
```

Arguments

`proc.type`	Type of simulation, `"indep"` for independent chains at the thermodynamic states, `"parallel"` for (synchronous) parallel tempering, or `"serial"` for (single-chain) serial tempering (see the details).
`block.size`	A vector of length `m=sum(size>0)` (or recycled to be so), giving possibly different block sizes for the `m` sampled thermodynamic states if `proc.type="indep"`, or a scalar (or truncated to the first element), giving a single block size if `proc.type="parallel"` or `"serial"`.
`boot.size`	The number of bootstrap replications.
`seed`	Seed for random number generation.
`label`	A vector of length `N` of labels between `1` to `M` such that `label[i]=j` when `i`th observation is obtained from `j`th thermodynamic state; either `label` or `size` must be provided if `proc.type="indep"` or `"parallel"`; `label` must be provided but `size` is optional if `proc.type="serial"`.

logQ	N x M matrix of log unnormalized densities (such as $1/kT$ times negative potential energies), where N is the total sample size, i.e., sum(size), and M is the number of thermodynamic states for which free energies are to be computed; the ith row of logQ correspond to ith observations and the jth column correspond to jth thermodynamic state.
size	A vector of length M, giving the individual sample sizes for the M thermodynamic states, ordered as the columns of logQ; if NULL, then label is required and used to compute size.
base	The baseline index, between 1 to M, for the thermodynamic state (with sample size >0) whose free energy is set to 0; if NULL, then base is set to the first index j such that size[j]>0.
init	A vector of length M, giving the initial values of the log-normalizing constants (or log of the partition functions); if NULL, then init is set to the zero vector.
phi	A vector of function values on the pooled sample; if NULL, no expectation is estimated.
state	A vector of indices for the thermodynamic states under which expectations are to be computed; if NULL, no expectation is estimated.

Details

The use of block bootstrap requires at least two more inputs than uwham: the type of simulation proc.type and the time ordering of the observations.

If proc.type="indep", the data are assumed to be generated by (approximately) independent chains at different thermodynamic states. The observations corresponding to the rows of logQ are assumed to be ordered by thermodynamic state and by simulation time within each state. To perform block bootstrap, data blocks are resampled within each thermodynamic state and then pooled to build bootstrap samples.

If proc.type="parallel", the data are assumed to be generated by (synchronous) parallel tempering or replica exhanges. Equal sample sizes are required. The observations corresponding to the rows of logQ are assumed to be ordered by simulation time and then by *either* thermodynamic state (as in the dataset ligand2.soft) *or* replica (as in the dataset ligand2.hard). The synchronized block bootstrap is implemented such that blocks corresponding to the same time interval are randomly selected from all the thermodynamic states (Tan et al. 2012).

If proc.type="serial", the data are assumed to be generated by (single-chain) serial tempering or simulated tempering. The observations corresponding to the rows of logQ are assumed to be ordered by time. The thermodynamic labels must then be provided in label. To perform block bootstrap, data blocks are resampled from the entire chain, and the number of observations at each state is re-computed for each bootstrap sample.

Value

ze	The vector of averages of estimated free energies over the bootstrap samples, which can differ from the output ze in uwham.
ve	The vector of estimated variances for the output ze in uwham.
phi	The vector of averages of estimated expectations over the bootstrap samples (which can differ from the output phi in uwham.phi), if phi!=NULL and state!=NULL.

phi.v	The vector of estimated variances for phi in uwham.phi, if phi!=NULL and state!=NULL.

References

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

Examples

```
#See the examples for uwham().
```

---

uwham.phi *Unbinned weighted histogram analysis method (UWHAM) for estimating expectations*

Description

This function implements UWHAM for estimating expectations for multiple distributions (such as multiple generalized ensembles) as described in Tan et al. (2012).

Usage

```
uwham.phi(phi, state, out.uwham, fisher = TRUE)
```

Arguments

phi	A vector of function (or observable) values on the pooled sample.
state	A vector of indices between 1 to M for the thermodynamic states under which expectations are to be computed.
out.uwham	The output of uwham.
fisher	Logical; if NULL, no variance estimation; if TRUE, variance estimation is based on Fisher information; if FALSE variance estimation is based on the Sandwich variance formula (see the details).

Details

The implementation is directly based on Tan et al. (2012). See the details for uwham.

Value

phi	The vector of estimated expectations.
phi.v	The vector of estimated variances for phi, if fisher!=NULL.
phi.V	The variance-covariance matrix for phi, if fisher!=NULL.

References

Tan, Z., Gallicchio, E., Lapelosa, M., and Levy, R.M. (2012) "Theory of binless multi-state free energy estimation with applications to protein-ligand binding," *Journal of Chemical Physics*, 136, 144102.

Examples

```
#See the examples for uwham().
```

Index