Package 'BSL'

January 16, 2019

Type Package

Title Bayesian Synthetic Likelihood

Version 2.0.0

Date 2019-01-15

Description
Bayesian synthetic likelihood (BSL, Price et al. (2018) <doi:10.1080/10618600.2017.1302882>) is an alternative to standard, non-parametric approximate Bayesian computation (ABC). BSL assumes a multivariate normal distribution for the summary statistic likelihood and it is suitable when the distribution of the model summary statistics is sufficiently regular. This package provides a Metropolis Hastings Markov chain Monte Carlo implementation of BSL, BSLasso and semiBSL. BSL with graphical lasso (BSLasso, An et al. (2018) <https://eprints.qut.edu.au/102263/>) is computationally more efficient when the dimension of the summary statistic is high. A semi-parametric version of BSL (semiBSL, An et al. (2018) <arXiv:1809.05800>) is more robust to non-normal summary statistics. Extensions to this package are planned.

Depends R (>= 3.4.0)

License GPL (>= 2)

LazyLoad yes

Imports glasso, ggplot2, MASS, mvtnorm, copula, cvTools, graphics, gridExtra, foreach, coda, Rcpp, methods

Suggests elliplot, doParallel

LinkingTo Rcpp, RcppArmadillo

LazyData true

RoxygenNote 6.1.1

Encoding UTF-8

Collate 'BSL_package.R' 'RcppExports.R' 'S3_penbsl.R' 'S4Functions.R' 'S4.R' 'bsl.R' 'cell.R' 'combinePlotsBSL.R' 'covWarton.R' 'gaussianSynLike.R' 'glasso_cv.R' 'imports.R' 'kernelCDF.R' 'logitTransform.R' 'ma2.R' 'multignk.R' 'selectPenalty.R' 'semiparaKernelEstimate.R'

NeedsCompilation yes

Author Ziwen An [aut, cre] (<https://orcid.org/0000-0002-9947-5182>),
Leah F. South [aut] (<https://orcid.org/0000-0002-5646-2963>),
Christopher C. Drovandi [aut] (<https://orcid.org/0000-0001-9222-8763>)

Maintainer Ziwen An <ziwen.an@hdr.qut.edu.au>

Repository CRAN

Date/Publication 2019-01-16 08:40:19 UTC

R topics documented:

BSL-package 2
bsl 3
bsl-class 6
cell 9
combinePlotsBSL 12
gaussianSynLike 14
ma2 15
mgnk 18
penbsl 21
plot.bsl 21
selectPenalty 23
semiparaKernelEstimate 25
show.bsl 26
summary.bsl 26

Index **28**

`BSL-package` *Bayesian synthetic likelihood*

Description

Bayesian synthetic likelihood (BSL, Price et al (2018)) is an alternative to standard, non-parametric approximate Bayesian computation (ABC). BSL assumes a multivariate normal distribution for the summary statistic likelihood and it is suitable when the distribution of the model summary statistics is sufficiently regular.

In this package, a Metropolis Hastings Markov chain Monte Carlo (MH-MCMC) implementation of BSL is available. We also include an implementation of BSLasso (An et al., 2018a), which is computationally more efficient when the dimension of the summary statistic is high, and semiBSL (An et al. 2018b), which relaxes the normality assumption to an extent and maintains the computational advantages of BSL without any tuning.

Parallel computing is supported through the `foreach` package and users can specify their own parallel backend by using packages like `doParallel` or `doMC`.

The main functionality is available through

- `bsl`: The general function to perform BSL, BSLasso and semiBSL (with or without parallel computing).
- `selectPenalty`: A function to select the penalty for BSLasso.

Several examples have also been included. These examples can be used to reproduce the results of An et al. (2018a).

- `ma2`: The MA(2) example from An et al. (2018a).
- `mgnk`: The multivariate G&K example from An et al. (2018a).
- `cell`: The cell biology example from Price et al. (2018) and An et al. (2018a).

Extensions to this package are planned.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

References

Price, L. F., Drovandi, C. C., Lee, A., & Nott, D. J. (2018). Bayesian synthetic likelihood. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2017.1302882

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018a). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

An, Z., Nott, D. J. & Drovandi, C. (2018b). Robust Bayesian Synthetic Likelihood via a Semi-Parametric Approach. ArXiv Preprint https://arxiv.org/abs/1809.05800

`bsl` *Performing BSL, BSLasso and semiBSL*

Description

This is the main function for performing MCMC BSL, MCMC BSLasso and MCMC semiBSL. Parallel computing is supported with the R package `foreach`.

Usage

```
bsl(y, n, M, theta0, covRandWalk, fnSim, fnSum, method = c("BSL",
  "semiBSL")[1], shrinkage = NULL, penalty = NULL, fnPrior = NULL,
  simArgs = NULL, sumArgs = NULL, logitTransformBound = NULL,
  standardise = FALSE, parallel = FALSE, parallelArgs = NULL,
  thetaNames = NULL, plotOnTheFly = FALSE, verbose = FALSE)
```

Arguments

y	The observed data - note this should be the raw dataset NOT the set of summary statistics.
n	The number of simulations from the model per MCMC iteration for estimating the synthetic likelihood.
M	The number of MCMC iterations.
theta0	Initial guess of the parameter value, which is used as the starting value for MCMC.
covRandWalk	A covariance matrix to be used in multivariate normal random walk proposals.
fnSim	A function that simulates data for a given parameter value. The first argument should be the parameters. Other necessary arguments (optional) can be specified with simArgs.
fnSum	A function for computing summary statistics of data. The first argument should be the observed or simulated dataset. Other necessary arguments (optional) can be specified with sumArgs.
method	A string argument indicating the method to be used. The default, 'BSL', runs standard BSL or BSLasso if shrinkage is used. 'semiBSL' runs the semi-parametric BSL algorithm and is more robust to non-normal summary statistics.
shrinkage	A string argument indicating which shrinkage method to be used. The default is NULL, which means no shrinkage is used. Current options are 'glasso' for graphical lasso and 'Warton' for the ridge regularisation method of Warton (2008).
penalty	The penalty value to be used for the specified shrinkage method. Must be between zero and one if the shrinkage method is 'Warton'.
fnPrior	A function that computes the prior density for a parameter. The default is NULL, which is an improper flat prior over the real line for each parameter. The function must have a single input: a vector of parameter values.
simArgs	A list of additional arguments to pass into the simulation function. Only use when the input fnSim requires additional arguments. The default is NULL.
sumArgs	A list of additional arguments to pass into the summary statistics function. Only use when the input fnSum requires additional arguments. The default is NULL.
logitTransformBound	A p by 2 numeric matrix indicating the upper and lower bound of parameters if a logit transformation is used on the parameter space, where p is the number of parameters. The default is NULL, which means no logit transformation is used. It is also possible to define other transformations with fnSim and fnPrior. The first column contains the lower bound of each parameter and the second column contains the upper bound. Infinite lower or upper bound is also supported, eg. matrix(c(1,Inf,0,10,-Inf,0.5),3,2,byrow=TRUE).
standardise	A logical argument that determines whether to standardise the summary statistics before applying the graphical lasso. This is only valid if shrinkage is 'glasso' and penalty is not NULL. The diagonal elements will not be penalised if the shrinkage method is 'glasso'. The default is FALSE.

parallel	A logical value indicating whether parallel computing should be used for simulation and summary statistic evaluation. The default is FALSE. When model simulation is fast, it may be preferable to perform serial computations to avoid significant communication overhead between workers.
parallelArgs	A list of additional arguments to pass into the foreach function. Only used when parallel computing is enabled, default is NULL.
thetaNames	A string vector of parameter names, which must have the same length as the parameter vector. The default is NULL.
plotOnTheFly	A logical argument. If TRUE, a plot of approximate univariate posteriors based on the current accepted samples will be shown every 1000 iterations. The default is FALSE.
verbose	A logical argument indicating whether the iteration numbers (1:M) and accepted proposal flags should be printed to track progress. The default is FALSE.

Value

An object of class bsl is returned, containing the following components:

- theta: MCMC samples from the joint approximate posterior distribution of the parameters.
- loglike: Accepted MCMC samples of the estimated log-likelihood values.
- acceptanceRate: The acceptance rate of the MCMC algorithm.
- earlyRejectionRate: The early rejection rate of the algorithm (early rejection may occur when using bounded prior distributions).
- call: The original code that was used to call the method.
- y: The input observed data.
- n: The input number of simulations from the model per MCMC iteration.
- M: The input number of MCMC iterations.
- theta0: The input initial guess of the parameter value.
- covRandWalk: The input covariance matrix used in multivariate normal random walk proposals.
- fnSim: The input data simulation function.
- fnSum: The input function for computing summary statistics of data.
- method: The input string argument indicating the used method.
- shrinkage: The input string argument indicating the shrinkage method.
- penalty: The input penalty value.
- fnPrior: The input function that computes the prior density for a parameter.
- simArgs: The input list of additional arguments to pass into the simulation function.
- sumArgs: The input list of additional arguments to pass into the summary statistics function.
- logitTransform: The logical argument indicating whether a logit transformation is used in the algorithm.
- logitTransformBound: The input matrix of logitTransformBound.

- `standardise`: The input logical argument that determines whether to standardise the summary statistics.
- `parallel`: The input logical value indicating whether parallel computing is used in the process.
- `parallelArgs`: The input list of additional arguments to pass into the `foreach` function.
- `thetaNames`: The string vector of parameter names.
- `time`: The running time of class `difftime`.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

References

Price, L. F., Drovandi, C. C., Lee, A., & Nott, D. J. (2018). Bayesian synthetic likelihood. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2017.1302882

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018a). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

An, Z., Nott, D. J. & Drovandi, C. (2018b). Robust Bayesian Synthetic Likelihood via a Semi-Parametric Approach. ArXiv Preprint https://arxiv.org/abs/1809.05800

Warton, D. I. (2008). Penalized Normal Likelihood and Ridge Regularization of Correlation and Covariance Matrices, Journal of the American Statistical Association. https://doi.org/10.1198/016214508000000021

See Also

`selectPenalty` for a function to tune the BSLasso tuning parameter and `plot` for functions related to visualisation.

`bsl-class` *S4 class* `"bsl"`.

Description

S4 class `"bsl"` for Results of `bsl` Function.

Summarise a bsl class object.

Plot the univariate marginal posterior plot of a bsl class object.

Usage

```
## S4 method for signature 'bsl'
show(object)

summary(object, ...)

## S4 method for signature 'bsl'
summary(object, thetaNames = NULL)

plot(x, y, ...)

## S4 method for signature 'bsl,missing'
plot(x, which = 1L, thin = 1,
  thetaTrue = NULL, options.plot = NULL,
  top = "Approximate Univariate Posteriors", options.density = list(),
  options.theme = list())
```

Arguments

object	A "bsl" class object to be displayed.
...	Other arguments.
thetaNames	Parameter names to be shown in the summary table. Parameter names of the bsl object will be used by default.
x	A "bsl" class object to plot.
y	Ignore.
which	An integer argument indicating which plot function to be used. The default, 1L, uses the plain plot to visualise the result. 2L uses ggplot2 to generate an aesthetically nicer figure.
thin	A numeric argument indicating the gap between samples to be taken when thinning the MCMC draws. The default is 1L, which means no thinning is used.
thetaTrue	A set of values to be included on the plots as a reference line. The default is NULL.
options.plot	A list of additional arguments to pass into the plot function. Only use when which is 1L.
top	A character argument of the combined plot title if which is 2L.
options.density	A list of additional arguments to pass into the geom_density function. Only use when which is 2L.
options.theme	A list of additional arguments to pass into the theme function. Only use when which is 2L.

Value

A vector of the number of simulations per iteration, acceptance rate of the Markov chain annd scaled effective sample size for each parameter.

Methods (by generic)

- show: Display the basic information of a bsl object. See show.bsl.
- summary: Summarise a bsl class object. See summary.bsl.
- plot: Plot the univariate marginal posterior plot of a bsl class object. See plot.bsl.

Slots

theta Object of class "matrix". MCMC samples from the joint approximate posterior distribution of the parameters.

loglike Object of class "numeric". Accepted MCMC samples of the estimated log-likelihood values.

acceptanceRate Object of class "numeric". The acceptance rate of the MCMC algorithm.

earlyRejectionRate Object of class "numeric". The early rejection rate of the algorithm (early rejection may occur when using bounded prior distributions).

call Object of class "call". The original code that was used to call the method.

y Object of class "ANY". The observed data.

n Object of class "numeric". The number of simulations from the model per MCMC iteration.

M Object of class "numeric". The number of MCMC iterations.

theta0 Object of class "numeric". The initial guess of the parameter value.

covRandWalk Object of class "matrix". The covariance matrix used in multivariate normal random walk proposals.

fnSim Object of class "function". The data simulation function.

fnSum Object of class "function". The function for computing summary statistics of data.

method Object of class "character". The character argument indicating the used method.

shrinkage Object of class "characterOrNULL". The character argument indicating the shrinkage method.

penalty Object of class "numericOrNULL". The penalty value.

fnPrior Object of class "functionOrNULL". The function that computes the prior density for a parameter.

simArgs Object of class "listOrNULL". The list of additional arguments to pass into the simulation function.

sumArgs Object of class "listOrNULL". The list of additional arguments to pass into the summary statistics function.

logitTransform Object of class "logical". The logical argument indicating whether a logit transformation is used in the algorithm.

logitTransformBound Object of class "matrixOrNULL". The matrix of logitTransformBound.

standardise Object of class "logical". The logical argument that determines whether to standardise the summary statistics.

parallel Object of class "logical". The logical value indicating whether parallel computing is used in the process.

`parallelArgs` Object of class `"listOrNULL"`. The list of additional arguments to pass into the `foreach` function.

`thetaNames` Object of class `"expression"`. The character vector of parameter names.

`time` Object of class `"difftime"`. The running time.

`cell` *Cell biology example*

Description

This example estimates the probabilities of cell motility and cell proliferation for a discrete-time stochastic model of cell spreading. We provide the data and tuning parameters required to reproduce the results in An et al. (2018).

Usage

```
cell_sim(theta, Yinit, rows, cols, sim_iters, num_obs)

cell_sum(Y, Yinit)

cell_prior(theta)
```

Arguments

`theta`	A vector of proposed model parameters, Pm and Pp.
`Yinit`	The initial matrix of cell presences of size `rows` x `cols`.
`rows`	The number of rows in the lattice (rows in the cell location matrix).
`cols`	The number of columns in the lattice (columns in the cell location matrix).
`sim_iters`	The number of discretisation steps to get to when an observation is actually taken. For example, if observations are taken every 5 minutes but the discretisation level is 2.5 minutes, then `sim_iters` would be 2. Larger values of `sim_iters` lead to more "accurate" simulations from the model, but they also increase the simulation time.
`num_obs`	The total number of images taken after initialisation.
`Y`	A `rows` x `cols` x `num_obs` array of the cell presences at times `1:num_obs` (not time 0).

Details

Cell motility (movement) and proliferation (reproduction) cause tumors to spread and wounds to heal. If we can measure cell proliferation and cell motility under different situations, then we may be able to use this information to determine the efficacy of different medical treatments.

A common method for measuring in vitro cell movement and proliferation is the scratch assay. Cells form a layer on an assay and, once they are completely covering the assay, a scratch is made to separate the cells. Images of the cells are taken until the scratch has closed up and the cells are in

contact again. Each image can be converted to a binary matrix by forming a lattice and recording the binary matrix (of size `rows` x `cols`) of cell presences.

The model that we consider is a random walk model with parameters for the probability of cell movement (Pm) and the probability of cell proliferation (`Pp`) and it has no tractable likelihood function. We use the uninformative priors θ_1 $U(0, 1)$ and θ_2 $U(0, 1)$.

We have a total of 145 summary statistics, which are made up of the Hamming distances between the binary matrices for each time point and the total number of cells at the final time.

Details about the types of cells that this model is suitable for and other information can be found in Price et al. (2018) and An et al. (2018). Johnston et al. (2014) use a different ABC method and different summary statistics for a similar example.

A simulated dataset

An example 'observed' dataset and the tuning parameters relevant to that example can be obtained using `data(cell)`. This 'observed' data is a simulated dataset with $Pm = 0.35$ and $Pp = 0.001$. The lattice has 27 `rows` and 36 `cols` and there are `num_obs = 144` observations after time 0 (to mimic images being taken every 5 minutes for 12 hours). The simulation is based on there initially being 110 cells in the assay.

Further information about the specific choices of tuning parameters used in BSL and BSLasso can be found in An et al. (2018).

- `data`: The `rows` x `cols` x `num_obs` array of the cell presences at times 1:144.
- `sim_options`: Values of `sim_options` relevant to this example.
- `sum_options`: Values of `sim_options` relevant to this example, i.e. just the value of `Yinit`.
- `start`: A vector of suitable initial values of the parameters for MCMC.
- `cov`: Covariance matrix of the multivariate normal random walk, in the form of a $2x2$ matrix.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

References

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

Johnston, S., Simpson, M. J., McElwain, D. L. S., Binder, B. J. & Ross, J. V. (2014). Interpreting Scratch Assays Using Pair Density Dynamic and Approximate Bayesian Computation. Open Biology, 4, 1-11.

Price, L. F., Drovandi, C. C., Lee, A., & Nott, D. J. (2018). Bayesian synthetic likelihood. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2017.1302882

Examples

```
require(doParallel) # You can use a different package to set up the parallel backend

# Loading the data for this example
data(cell)
true_cell <- c(0.35, 0.001)

# Performing BSL (reduce the number of iterations M if desired)
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
resultCellBSL <- bsl(cell$data, n = 5000, M = 10000, theta0 = cell$start, covRandWalk = cell$cov,
                     fnSim = cell_sim, fnSum = cell_sum, fnPrior = cell_prior,
                     simArgs = cell$sim_options, sumArgs = cell$sum_options,
                     parallel = TRUE, parallelArgs = list(.packages = 'BSL'),
                     thetaNames = expression(P[m], P[p]), verbose = TRUE)
stopCluster(cl)
registerDoSEQ()
show(resultCellBSL)
summary(resultCellBSL)
plot(resultCellBSL, thetaTrue = true_cell, thin = 20)

# Performing tuning for BSLasso
lambda_all <- list(exp(seq(0.5,2.5,length.out=20)), exp(seq(0,2,length.out=20)),
                   exp(seq(-1,1,length.out=20)), exp(seq(-1,1,length.out=20)))
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
sp_cell <- selectPenalty(ssy = cell_sum(cell$data, cell$sum_options$Yinit),
                         n = c(500, 1000, 1500, 2000), lambda_all, theta = true_cell,
                         M = 100, sigma = 1.5, fnSim = cell_sim,
                         fnSum = cell_sum, simArgs = cell$sim_options,
                         sumArgs = cell$sum_options, parallelSim = TRUE,
                         parallelSimArgs = list(.packages = 'BSL'), parallelMain = TRUE)
stopCluster(cl)
registerDoSEQ()
sp_cell
plot(sp_cell)

# Performing BSLasso with a fixed penalty (reduce the number of iterations M if desired)
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
resultCellBSLasso <- bsl(cell$data, n = 1500, M = 10, theta0 = cell$start,
                         covRandWalk = cell$cov, fnSim = cell_sim, fnSum = cell_sum,
                         shrinkage = 'glasso', penalty = 1.3, fnPrior = cell_prior,
                         simArgs = cell$sim_options, sumArgs = cell$sum_options,
                         parallel = TRUE, parallelArgs = list(.packages = 'BSL'),
                         thetaNames = expression(P[m], P[p]), verbose = TRUE)
stopCluster(cl)
registerDoSEQ()
```

```
show(resultCellBSLasso)
summary(resultCellBSLasso)
plot(resultCellBSLasso, thetaTrue = true_cell, thin = 20)

# Performing semiBSL (reduce the number of iterations M if desired)
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
resultCellSemiBSL <- bsl(cell$data, n = 5000, M = 10000, theta0 = cell$start,
                         covRandWalk = cell$cov, fnSim = cell_sim, fnSum = cell_sum,
                         method = 'semiBSL', fnPrior = cell_prior,
                         simArgs = cell$sim_options, sumArgs = cell$sum_options,
                         parallel = TRUE, parallelArgs = list(.packages = 'BSL'),
                         thetaNames = expression(P[m], P[p]), verbose = TRUE)
stopCluster(cl)
registerDoSEQ()
show(resultCellSemiBSL)
summary(resultCellSemiBSL)
plot(resultCellSemiBSL, thetaTrue = true_cell, thin = 20)

# Plotting the results together for comparison
# plot using the R default plot function
par(mar = c(5, 4, 1, 2), oma = c(0, 1, 2, 0))
combinePlotsBSL(list(resultCellBSL, resultCellBSLasso, resultCellSemiBSL),
    which = 1, thetaTrue = true_cell, thin = 20, label = c('bsl', 'bslasso', 'semiBSL'),
    col = c('red', 'blue', 'green'), lty = 2:4, lwd = 1)
mtext('Approximate Univariate Posteriors', outer = TRUE, cex = 1.5)

# plot using the ggplot2 package
combinePlotsBSL(list(resultCellBSL, resultCellBSLasso, resultCellSemiBSL),
   which = 2, thetaTrue = true_cell, thin = 20, label = c('bsl   ', 'bslasso  ', 'semiBSL'),
    options.color = list(values=c('red', 'blue', 'green')),
    options.linetype = list(values = 2:4), options.size = list(values = rep(1, 3)),
    options.theme = list(plot.margin = grid::unit(rep(0.03,4),"npc"),
   axis.title = ggplot2::element_text(size=12), axis.text = ggplot2::element_text(size = 8),
    legend.text = ggplot2::element_text(size = 12)))
```

`combinePlotsBSL` *Plot the densities of multiple "bsl" class objects.*

Description

The function `combinePlotsBSL` can be used to plot multiple BSL densities together, optionally with the true values for the parameters.

Usage

```
combinePlotsBSL(objectList, which = 1L, thin = 1, thetaTrue = NULL,
  label = NULL, legendPosition = c("auto", "right", "bottom")[1],
  legendNcol = NULL, col = NULL, lty = NULL, lwd = NULL,
  cex.lab = 1, cex.axis = 1, cex.legend = 0.75,
  top = "Approximate Marginal Posteriors", options.color = list(),
  options.linetype = list(), options.size = list(),
  options.theme = list())
```

Arguments

objectList	A list of "bsl" class objects.
which	An integer argument indicating which plot function to be used. The default, 1L, uses the plain plot to visualise the result. 2L uses ggplot2 to generate an aesthetically nicer figure.
thin	A numeric argument indicating the gap between samples to be taken when thinning the MCMC draws. The default is 1L, which means no thinning is used.
thetaTrue	A set of values to be included on the plots as a reference line. The default is NULL.
label	A string vector indicating the labels to be shown in the plot legend. The default is NULL, which uses the names from objectList.
legendPosition	One of the three string arguments, "auto", "right" or "bottom", indicating the legend position. The default is "auto", which automatically choose from "right" and "bottom". Only used when which is 1L.
legendNcol	A integer argument indicating the number of columns of the legend. The default, NULL, put all legends in the same row or column depending on legendPosition. Only used when which is 1L.
col	A vector argument containing the plotting color for each density curve. Each element of the vector will be passed into lines. Only used when which is 1L.
lty	A vector argument containing the line type for each density curve. Each element of the vector will be passed into lines. Only used when which is 1L.
lwd	A vector argument containing the line width for each density curve. Each element of the vector will be passed into lines. Only used when which is 1L.
cex.lab	The magnification to be used for x and y labels relative to the current setting of cex. To be passed into plot. Only used when which is 1L.
cex.axis	The magnification to be used for axis annotation relative to the current setting of cex. To be passed into plot. Only used when which is 1L.
cex.legend	The magnification to be used for legend annotation relative to the current setting of cex. Only used when which is 1L.
top	A string argument of the combined plot title. Only used when which is 2L.
options.color	A list of additional arguments to pass into function ggplot2::scale_color_manual. Only used when which is 2L.
options.linetype	A list of additional arguments to pass into function ggplot2::scale_linetype_manual. Only used when which is 2L.

options.size	A list of additional arguments to pass into function ggplot2::scale_size_manual. Only used when which is 2L.
options.theme	A list of additional arguments to pass into the theme function. Only use when which is 2L.

See Also

ma2 for an example

---

gaussianSynLike *Estimating the Gaussian synthetic likelihood*

Description

This function estimates the Gaussian synthetic likelihood function of Wood (2010). Shrinkage on the Gaussian covariance matrix is also available (see An et al 2018).

Usage

```
gaussianSynLike(ssy, ssx, shrinkage = NULL, penalty = NULL,
  standardise = FALSE, log = TRUE, verbose = FALSE)
```

Arguments

ssy	The observed summary statisic.
ssx	A matrix of the simulated summary statistics. The number of rows is the same as the number of simulations per iteration.
shrinkage	A string argument indicating which shrinkage method to be used. The default is NULL, which means no shrinkage is used. Current options are 'glasso' for graphical lasso and 'Warton' for the ridge regularisation method of Warton (2008).
penalty	The penalty value to be used for the specified shrinkage method. Must be between zero and one if the shrinkage method is 'Warton'.
standardise	A logical argument that determines whether to standardise the summary statistics before applying the graphical lasso. This is only valid if shrinkage is 'glasso' and penalty is not NULL. The diagonal elements will not be penalised if the shrinkage method is 'glasso'. The default is FALSE.
log	A logical argument indicating if the log of likelihood is given as the result. The default is TRUE.
verbose	A logical argument indicating whether an error message should be printed if the function fails to compute a likelihood. The default is FALSE.

Value

The estimated (log) likelihood value.

References

Price, L. F., Drovandi, C. C., Lee, A., & Nott, D. J. (2018). Bayesian synthetic likelihood. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2017.1302882

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

Friedman, J., Hastie, T., Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics. https://doi.org/10.1093/biostatistics/kxm045

Warton, D. I. (2008). Penalized Normal Likelihood and Ridge Regularization of Correlation and Covariance Matrices, Journal of the American Statistical Association. https://doi.org/10.1198/016214508000000021

Examples

```
data(ma2)
y <- ma2$data # the observed data

# function that simulates an ma2 time series
simulate_ma2 <- function(theta, L = 50) {
    rand <- rnorm(L + 2)
    y <- rand[3 : (L+2)] + theta[1] * rand[2 : (L+1)] + theta[2] * rand[1 : L]
    return(y)
}

theta_true <- c(0.6, 0.2)
x <- matrix(0, 300, 50)
set.seed(100)
for(i in 1:300) x[i, ] <- simulate_ma2(theta_true)

# the standard Gaussian synthetic likelihood (the likelihood estimator used in BSL)
gaussianSynLike(y, x)
# the Gaussian synthetic likelihood with glasso shrinkage estimation
# (the likelihood estimator used in BSLasso)
gaussianSynLike(y, x, shrinkage = 'glasso', penalty = 0.1)
```

---

ma2 *An MA(2) model*

Description

In this example we wish to estimate the parameters of a simple MA(2) time series model. We provide the data and tuning parameters required to reproduce the results in An et al. (2018).

Usage

```
data(ma2)

ma2_sim(theta, T)

ma2_sum(x)

ma2_prior(theta)
```

Arguments

theta	A vector of proposed model parameters, θ_1 and θ_2
T	The number of observations.
x	Observed or simulated data in the format of a vector of length T.

Details

This example is based on estimating the parameters of a basic MA(2) time series model of the form

$$y_t = z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2},$$

where $t = 1, \ldots, T$ and z_t $N(0, 1)$ for $t = -1, 0, \ldots, T$. A uniform prior is used for this example, subject to the restrictions that $-2 < \theta_1 < 2$, $\theta_1 + \theta_2 > -1$ and $\theta_1 - \theta_2 < 1$ so that invertibility of the time series is satisfied. The summary statistics are simply the full data.

A simulated dataset

An example 'observed' dataset and the tuning parameters relevant to that example can be obtained using `data(ma2)`. This 'observed' data is a simulated dataset with $\theta_1 = 0.6$, $\theta_2 = 0.2$ and $T = 50$. Further information about this model and the specific choices of tuning parameters used in BSL and BSLasso can be found in An et al. (2018).

- `data`: A time series dataset, in the form of a vector of length T
- `sim_options`: A list containing $T = 50$
- `start`: A vector of suitable initial values of the parameters for MCMC
- `cov`: Covariance matrix of the multivariate normal random walk, in the form of a $2x2$ matrix

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

References

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

Examples

```
# Loading the data for this example
data(ma2)
true_ma2 <- c(0.6,0.2)

# Performing BSL (reduce the number of iterations M if desired)
resultMa2BSL <- bsl(y = ma2$data, n = 500, M = 300000, theta0 = ma2$start, covRandWalk = ma2$cov,
    fnSim = ma2_sim, fnSum = ma2_sum, fnPrior = ma2_prior, simArgs = ma2$sim_options,
    thetaNames = expression(theta[1], theta[2]), verbose = TRUE)
show(resultMa2BSL)
summary(resultMa2BSL)
plot(resultMa2BSL, which = 1, thetaTrue = true_ma2, thin = 20)

# Performing tuning for BSLasso
lambda_all <- list(exp(seq(-3,0.5,length.out=20)), exp(seq(-4,-0.5,length.out=20)),
                   exp(seq(-5.5,-1.5,length.out=20)), exp(seq(-7,-2,length.out=20)))
sp_ma2 <- selectPenalty(ssy = ma2_sum(ma2$data), n = c(50, 150, 300, 500), lambda_all,
                        theta = true_ma2, M = 100, sigma = 1.5, fnSim = ma2_sim,
                        fnSum = ma2_sum, simArgs = ma2$sim_options)
sp_ma2
plot(sp_ma2)

# Performing BSLasso with a fixed penalty (reduce the number of iterations M if desired)
resultMa2BSLasso <- bsl(y = ma2$data, n = 300, M = 250000, theta0 = ma2$start, covRandWalk=ma2$cov,
   fnSim = ma2_sim, fnSum = ma2_sum, shrinkage = 'glasso', penalty = 0.027, fnPrior = ma2_prior,
   simArgs = ma2$sim_options, thetaNames = expression(theta[1], theta[2]), verbose = TRUE)
show(resultMa2BSLasso)
summary(resultMa2BSLasso)
plot(resultMa2BSLasso, which = 1, thetaTrue = true_ma2, thin = 20)

# Performing semiBSL (reduce the number of iterations M if desired)
resultMa2SemiBSL <- bsl(y = ma2$data, n = 500, M = 300000, theta0 = ma2$start, covRandWalk=ma2$cov,
                    fnSim = ma2_sim, fnSum = ma2_sum, method = 'semiBSL', fnPrior = ma2_prior,
                    simArgs = ma2$sim_options, thetaNames = expression(theta[1], theta[2]),
                        verbose = TRUE)
show(resultMa2SemiBSL)
summary(resultMa2SemiBSL)
plot(resultMa2SemiBSL, which = 1, thetaTrue = true_ma2, thin = 20)

# Plotting the results together for comparison
# plot using the R default plot function
par(mar = c(5, 4, 1, 2), oma = c(0, 1, 2, 0))
combinePlotsBSL(list(resultMa2BSL, resultMa2BSLasso, resultMa2SemiBSL), which = 1,
    thetaTrue = true_ma2, thin = 20, label = c('bsl', 'bslasso', 'semiBSL'),
    col = c('red', 'blue', 'green'), lty = 2:4, lwd = 1)
mtext('Approximate Univariate Posteriors', outer = TRUE, cex = 1.5)

# plot using the ggplot2 package
combinePlotsBSL(list(resultMa2BSL, resultMa2BSLasso, resultMa2SemiBSL), which = 2,
    thetaTrue = true_ma2, thin = 20, label = c('bsl   ', 'bslasso   ', 'semiBSL'),
    options.color = list(values=c('red', 'blue', 'green')),
```

```
 options.linetype = list(values = 2:4), options.size = list(values = rep(1, 3)),
 options.theme = list(plot.margin = grid::unit(rep(0.03,4), "npc"),
axis.title = ggplot2::element_text(size=12), axis.text = ggplot2::element_text(size = 8),
 legend.text = ggplot2::element_text(size = 12)))
```

---

mgnk *The multivariate G&K example*

Description

Here we provide the data and tuning parameters required to reproduce the results from the multivariate G & K (Drovandi and Pettitt, 2011) example from An et al. (2018).

Usage

```
mgnk_sim(theta_tilde, T, J, bound)

mgnk_sum(y)
```

Arguments

theta_tilde	A vector with 15 elements for the proposed model parameters.
T	The number of observations in the data.
J	The number of variables in the data.
bound	A matrix of boundaries for the uniform prior.
y	A T x J matrix of data.

Details

It is not practical to give a reasonable explanation of this example through R documentation given the number of equations involved. We refer the reader to the BSLasso paper (An et al., 2018) at https://doi.org/10.1080/10618600.2018.1537928 for information on the model and summary statistic used in this example.

An example dataset

We use the foreign currency exchange data available from http://www.rba.gov.au/statistics/historical-data.html as in An et al. (2018).

- data: A 1651 x 3 matrix of data.
- sim_options: Values of sim_options relevant to this example.
- start: A vector of suitable initial values of the parameters for MCMC.
- cov: Covariance matrix of the multivariate normal random walk, in the form of a $15x15$ matrix.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

References

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

Drovandi, C. C. and Pettitt, A. N. (2011). Likelihood-free Bayesian estimation of multivariate quantile distributions. Computational Statistics and Data Analysis, 55(9):2541-2556.

Examples

```
require(doParallel) # You can use a different package to set up the parallel backend
require(MASS)
require(elliplot)

# Loading the data for this example
data(mgnk)

# Performing BSL (reduce the number of iterations M if desired)
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
resultMgnkBSL <- bsl(mgnk$data, n = 60, M = 80000, theta0 = mgnk$start, covRandWalk = mgnk$cov,
    fnSim = mgnk_sim, fnSum = mgnk_sum, simArgs = mgnk$sim_options,
   parallel = TRUE, parallelArgs = list(.packages = c('BSL', 'MASS'), .export = 'ninenum'),
    thetaNames = expression(a[1],b[1],g[1],k[1],a[2],b[2],g[2],k[2],a[3],b[3],g[3],k[3],
        delta[12],delta[13],delta[23]), verbose = TRUE)
stopCluster(cl)
registerDoSEQ()
show(resultMgnkBSL)
summary(resultMgnkBSL)
plot(resultMgnkBSL, which = 2, thin = 20)

# Performing tuning for BSLasso
lambda_all <- list(exp(seq(-2.5,0.5,length.out=20)), exp(seq(-2.5,0.5,length.out=20)),
                   exp(seq(-4,-0.5,length.out=20)), exp(seq(-5,-2,length.out=20)))

# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
sp_mgnk <- selectPenalty(ssy = mgnk_sum(mgnk$data), n = c(15, 20, 30, 50), lambda_all,
    theta = mgnk$start, M = 100, sigma = 1.5, fnSim = mgnk_sim,
    fnSum = mgnk_sum, simArgs = mgnk$sim_options, standardise = TRUE,
   parallelSim = TRUE, parallelSimArgs = list(.packages = c('BSL', 'MASS'), .export = 'ninenum'),
    parallelMain = TRUE)
stopCluster(cl)
registerDoSEQ()
sp_mgnk
```

```
plot(sp_mgnk)

# Performing BSLasso with a fixed penalty (reduce the number of iterations M if desired)
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
resultMgnkBSLasso <- bsl(mgnk$data, n = 20, M = 80000, theta0 = mgnk$start, covRandWalk = mgnk$cov,
     fnSim = mgnk_sim, fnSum = mgnk_sum, simArgs = mgnk$sim_options,
     penalty = 0.3, standardise = TRUE, parallel = TRUE,
     parallelArgs = list(.packages = c('BSL', 'MASS'), .export = 'ninenum'),
    thetaNames = expression(a[1],b[1],g[1],k[1],a[2],b[2],g[2],k[2],a[3],b[3],g[3],k[3],
         delta[12],delta[13],delta[23]), verbose = TRUE)
stopCluster(cl)
registerDoSEQ()
show(resultMgnkBSLasso)
summary(resultMgnkBSLasso)
plot(resultMgnkBSLasso, which = 2, thin = 20)


# Performing semiBSL (reduce the number of iterations M if desired)
# Opening up the parallel pools using doParallel
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)
resultMgnkSemiBSL <- bsl(mgnk$data, n = 60, M = 80000, theta0 = mgnk$start, covRandWalk = mgnk$cov,
    fnSim = mgnk_sim, fnSum = mgnk_sum, simArgs = mgnk$sim_options, method = 'semiBSL',
   parallel = TRUE, parallelArgs = list(.packages = c('BSL', 'MASS'), .export = 'ninenum'),
    thetaNames = expression(a[1],b[1],g[1],k[1],a[2],b[2],g[2],k[2],a[3],b[3],g[3],k[3],
        delta[12],delta[13],delta[23]), verbose = TRUE)
stopCluster(cl)
registerDoSEQ()
show(resultMgnkSemiBSL)
summary(resultMgnkSemiBSL)
plot(resultMgnkSemiBSL, which = 2, thin = 20)

# Plotting the results together for comparison
# plot using the R default plot function
par(mar = c(4, 4, 1, 1), oma = c(0, 1, 2, 0))
combinePlotsBSL(list(resultMgnkBSL, resultMgnkBSLasso, resultMgnkSemiBSL), which = 1, thin = 20,
   label = c('bsl', 'bslasso', 'semiBSL'), col = c('red', 'blue', 'green'), lty = 2:4, lwd = 1)
mtext('Approximate Univariate Posteriors', outer = TRUE, line = 0.75, cex = 1.2)

# plot using the ggplot2 package
combinePlotsBSL(list(resultMgnkBSL, resultMgnkBSLasso, resultMgnkSemiBSL), which = 2, thin = 20,
    label=c('bsl   ','bslasso   ','semiBSL'),
    options.color=list(values=c('red','blue','green')),
    options.linetype = list(values = 2:4), options.size = list(values = rep(1, 3)),
    options.theme = list(plot.margin = grid::unit(rep(0.03,4),"npc"),
   axis.title = ggplot2::element_text(size=12), axis.text = ggplot2::element_text(size = 8),
    legend.text = ggplot2::element_text(size = 12)))
```

penbsl — *S3 reference class of the result from tuning to select the optimal penalty for BSLasso*

Description

Two functions (print and plot) are provided for class "penbsl".

Usage

```
## S3 method for class 'penbsl'
print(x, digits = max(3L, getOption("digits") - 4L),
  ...)

## S3 method for class 'penbsl'
plot(x, logscale = TRUE, ...)
```

Arguments

x	A "penbsl" class object, typically the output of function selectPenalty.
digits	The number of digits to print.
...	Other arguments.
logscale	A logical indicator whether the x-axis (penalty) should be log transformed. The default is TRUE.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

plot.bsl — *Plot method for class "bsl"*

Description

Plot the univariate marginal posterior plot of a bsl class object.

Usage

```
plot.bsl(x, which = 1L, thin = 1, thetaTrue = NULL,
  options.plot = NULL, top = "Approximate Univariate Posteriors",
  options.density = list(), options.theme = list())

marginalPostDefault(x, thin = 1, thetaTrue = NULL,
  options.plot = NULL)
```

```
marginalPostGgplot(x, thin = 1, thetaTrue = NULL,
  top = "Approximate Univariate Posteriors", options.density = list(),
  options.theme = list())
```

Arguments

x	A "bsl" class object to plot.
which	An integer argument indicating which plot function to be used. The default, 1L, uses the plain plot to visualise the result. 2L uses ggplot2 to generate an aesthetically nicer figure.
thin	A numeric argument indicating the gap between samples to be taken when thinning the MCMC draws. The default is 1L, which means no thinning is used.
thetaTrue	A set of values to be included on the plots as a reference line. The default is NULL.
options.plot	A list of additional arguments to pass into the plot function. Only use when which is 1L.
top	A string argument of the combined plot title if which is 2L.
options.density	A list of additional arguments to pass into the geom_density function. Only use when which is 2L.
options.theme	A list of additional arguments to pass into the theme function. Only use when which is 2L.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

See Also

combinePlotBSL for a function to plot multiple BSL densities.

Examples

```
## Not run:
# pretend we had a bsl result
result <- new('bsl')
result@theta <- MASS::mvrnorm(10000, c(0.6, 0.2), diag(c(1, 1)))
result@M <- 10000

# plot using the R default plot function
par(mar = c(5, 4, 1, 2), oma = c(0, 1, 3, 0))
plot(result, which = 1, thin = 10, thetaTrue = c(0.6, 0.2),
     options.plot = list(cex.main = 1, col = 'red', lty = 2, lwd = 2, main = NA))
mtext('Approximate Univariate Posteriors', outer = TRUE, cex = 1.5)

# plot using the ggplot2 package
plot(result, which = 2, thin = 10, thetaTrue = c(0.6, 0.2),
     options.density = list(colour = 'darkblue', fill = 'grey80', size = 1),
     options.theme = list(plot.margin = grid::unit(rep(0.05,4), "npc"),
```

```
                          axis.text = ggplot2::element_text(size = 10)))

## End(Not run)
```

`selectPenalty` *Selecting BSLasso Penalty*

Description

This is the main function for selecting the penalty for BSLasso based on a point estimate of the parameters. Parallel computing is supported with the R package `foreach`.

Usage

```
selectPenalty(ssy, n, lambda_all, theta, M, sigma, fnSim, fnSum,
  simArgs = NULL, sumArgs = NULL, standardise = FALSE,
  parallelSim = FALSE, parallelSimArgs = NULL, parallelMain = FALSE,
  verbose = TRUE)
```

Arguments

`ssy`	A summary statistic vector for the observed data.
`n`	A vector of possible values of `n`, the number of simulations from the model per MCMC iteration for estimating the synthetic likelihood.
`lambda_all`	A list, with each entry containing the vector of penalty values to test for the corresponding choice of `n`.
`theta`	A point estimate of the parameter value which all of the simulations will be based on.
`M`	The number of repeats to use in estimating the standard deviation of the estimated log synthetic likelihood.
`sigma`	The standard deviation of the log synthetic likelihood estimator to aim for.
`fnSim`	A function that simulates data for a given parameter value. The first argument should be the parameters. Other necessary arguments (optional) can be specified with `simArgs`.
`fnSum`	A function for computing summary statistics of data. The first argument should be the observed or simulated dataset. Other necessary arguments (optional) can be specified with `sumArgs`.
`simArgs`	A list of additional arguments to pass into the simulation function. Only use when the input `fnSim` requires additional arguments. The default is `NULL`.
`sumArgs`	A list of additional arguments to pass into the summary statistics function. Only use when the input `fnSum` requires additional arguments. The default is `NULL`.

`standardise` A logical argument that determines whether to standardise the summary statistics before applying the graphical lasso. This is only valid if shrinkage is 'glasso' and penalty is not `NULL`. The diagonal elements will not be penalised if the shrinkage method is 'glasso'. The default is `FALSE`.

`parallelSim` A logical value indicating whether parallel computing should be used for simulation and summary statistic evaluation. Default is `FALSE`.

`parallelSimArgs` A string vector of package names to pass into the `foreach` function as argument '.package'. Only used when `parallel_sim` is `TRUE`, default is `NULL`.

`parallelMain` A logical value indicating whether parallel computing should be used to computing the graphical lasso function. Default is `FALSE`.

`verbose` A logical argument indicating whether the iteration numbers (`1:M`) should be printed to track progress. The default is `FALSE`.

Value

An object of class `penbsl` is returned, containing the following components:

- `resultsDF`: A data frame containing the following:
  - `n`: The choices of `n` that were specified.
  - `penalty`: The choices of the penalty that were specified.
  - `sigma`: The standard deviation of the log synthetic likelihood estimator under the above choices.
  - `sigmaOpt`: An indicator of whether it was the closest `sigma` to the desired one for each choice of `n`.
- `call`: The original code that was used to call the method.

The functions print() and plot() are both available for types of class `penbsl`.

Author(s)

Ziwen An, Leah F. South and Christopher C. Drovandi

References

An, Z., South, L. F., Nott, D. J. & Drovandi, C. C. (2018). Accelerating Bayesian synthetic likelihood with the graphical lasso. Journal of Computational and Graphical Statistics. https://doi.org/10.1080/10618600.2018.1537928

See Also

`bsl` for a function to run BSLasso after selecting the tuning parameter and `penbsl` for functions related to visualisation.

`semiparaKernelEstimate` *Estimating the semi-parametric joint likelihood*

Description

This function estimates the semi-parametric likelihood of An et al (2018). Kernel density estimates are used for modelling each univariate marginal distribution, and the dependence structure between summaries are captured using a Gaussian copula.

Usage

```
semiparaKernelEstimate(ssy, ssx, kernel = "gaussian", shrinkage = NULL,
  penalty = NULL, log = TRUE)
```

Arguments

`ssy`	The observed summary statistic.
`ssx`	A matrix of the simulated summary statistics. The number of rows is the same as the number of simulations per iteration.
`kernel`	A string argument indicating the smoothing kernel to pass into `density` for estimating the marginal distribution of each summary statistic. Only "gaussian" and "epanechnikov" are available. The default is "gaussian".
`shrinkage`	A string argument indicating which shrinkage method to be used on the correlation matrix of the Gaussian copula. The default is `NULL`, which means no shrinkage is used. Current options are 'glasso' for graphical lasso and 'Warton' for the ridge regularisation method of Warton (2008).
`penalty`	The penalty value to be used for the specified shrinkage method. Must be between zero and one if the shrinkage method is 'Warton'.
`log`	A logical argument indicating if the log of the likelihood is given as the result. The default is `TRUE`.

Value

The estimated (log) likelihood value.

References

An, Z., Nott, D. J. & Drovandi, C. (2018). Robust Bayesian Synthetic Likelihood via a Semi-Parametric Approach. https://arxiv.org/abs/1809.05800

Friedman, J., Hastie, T., Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics. https://doi.org/10.1093/biostatistics/kxm045

Warton, D. I. (2008). Penalized Normal Likelihood and Ridge Regularization of Correlation and Covariance Matrices, Journal of the American Statistical Association. https://www.tandfonline.com/doi/abs/10.1198/016214508000000021

Examples

```
data(ma2)
y <- ma2$data # the observed data

# function that simulates an ma2 time series
simulate_ma2 <- function(theta, L = 50) {
    rand <- rnorm(L + 2)
    y <- rand[3 : (L+2)] + theta[1] * rand[2 : (L+1)] + theta[2] * rand[1 : L]
    return(y)
}

theta_true <- c(0.6, 0.2)
x <- matrix(0, 300, 50)
set.seed(100)
for(i in 1:300) x[i, ] <- simulate_ma2(theta_true)

# the default semi-parametric synthetic likelihood estimator of semiBSL
semiparaKernelEstimate(y, x)
# using shrinkage on the correlation matrix of the Gaussian copula is also possible
semiparaKernelEstimate(y, x, shrinkage = 'Warton', penalty = 0.6)
```

---

show.bsl *Show method for class "bsl". Display the basic information of a bsl object.*

Description

Display the basic information of a bsl object.

Usage

```
show.bsl(object)
```

Arguments

object	A "bsl" class object to be displayed.

---

summary.bsl *Summary method for class "bsl"*

Description

Summarise a bsl class object.

Usage

```
summary.bsl(object, thetaNames = NULL)
```

Arguments

object	A "bsl" class object to be summarised.
thetaNames	Parameter names to be shown in the summary table. Parameter names of the bsl object will be used by default.

Value

A vector of the number of simulations per iteration, acceptance rate of the Markov chain annd scaled effective sample size for each parameter.

Index

BSL *(*BSL-package*)*, 2
bsl, *3*, 3, *24*
bsl,bsl-method *(*bsl-class*)*, 6
bsl-class, 6
BSL-package, 2

cell, *3*, 9
cell_prior *(*cell*)*, 9
cell_sim *(*cell*)*, 9
cell_sum *(*cell*)*, 9
combinePlotBSL, *22*
combinePlotsBSL, 12

gaussianSynLike, 14

ma2, *3*, *14*, 15
ma2_prior *(*ma2*)*, 15
ma2_sim *(*ma2*)*, 15
ma2_sum *(*ma2*)*, 15
marginalPostDefault *(*plot.bsl*)*, 21
marginalPostGgplot *(*plot.bsl*)*, 21
mgnk, *3*, 18
mgnk_sim *(*mgnk*)*, 18
mgnk_sum *(*mgnk*)*, 18

penbsl, 21, *24*
plot, *6*
plot *(*bsl-class*)*, 6
plot,bsl,missing-method *(*bsl-class*)*, 6
plot.bsl, *8*, 21
plot.penbsl *(*penbsl*)*, 21
print.penbsl *(*penbsl*)*, 21

selectPenalty, *3*, *6*, *21*, 23
semiparaKernelEstimate, 25
show,bsl-method *(*bsl-class*)*, 6
show.bsl, *8*, 26
summary *(*bsl-class*)*, 6
summary,bsl-method *(*bsl-class*)*, 6
summary.bsl, *8*, 26