Package 'BayesMassBal'

July 7, 2020

Type Package

Title Bayesian Data Reconciliation of Separation Processes

Version 0.2.0

Author Scott Koermer <skoermer@vt.edu>

Maintainer Scott Koermer <skoermer@vt.edu>

Description Bayesian tools that can be used to reconcile, or mass balance, mass flow rate data collected from chemical or particulate separation processes aided by constraints governed by the conservation of mass.
Functions included in the package aid the user in organizing and constraining data, using Markov chain Monte Carlo methods to obtain samples from Bayesian models, and in computation of the marginal likelihood of the data, given a particular model, for model selection. Marginal likelihood is approximated by methods in Chib S (1995) <doi:10.2307/2291521>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports Rdpack, Matrix, MASS, pracma, tmvtnorm, LaplacesDemon, HDInterval

RdMacros Rdpack

Suggests knitr, rmarkdown, covr, spelling, tgp, testthat

VignetteBuilder knitr

URL https://github.com/skoermer/BayesMassBal

BugReports https://github.com/skoermer/BayesMassBal/issues

Language en-US

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-07 09:00:16 UTC

R topics documented:

BMB *Bayesian Mass Balance*

Description

Allows the user to specify the covariance structure for a Bayesian mass balance, simulates draws from reconciled masses and relevant covariance matrix, and approximates the log-marginal likelihood.

Usage

```
BMB(
  X,
  y,
  cov.structure = c("indep", "component", "location"),
  priors = NA,
  BTE = c(500, 20000, 1),
  lml = FALSE,
  ybal = TRUE,
  verb = 1
)
```

Arguments

`X`	A matrix that maps constrained masses to observed masses. Can be built from the function `constrainProcess`, see documentation for details.
`y`	A list of matrices of observed mass flow rates. Each matrix is a separate sample component. The rows of each matrix index the sampling location, and the columns index the sample set number. Can be specified using the `importObservations` function.
`cov.structure`	Character string. `"indep"` allows for independent error. `"component"` indicates error for a single sample component, between locations, is correlated. `"location"` indicates error all observations at a single location, between sample components, is correlated. Not specifying `cov.structure` defaults to the `"indep"` structure.

`priors` Optional list of user specified hyperparameters for conjugate priors. When not specified, the function uses a set of default conjugate priors. To see the required list structure run `BMB` with `BTE = c(1,2,1)` and inspect the output. See Details for more information.

`BTE` Numeric vector giving `c(Burn-in,Total-iterations,and Every)` for MCMC approximation of target distributions. The function `BMB` produces a total number of samples of $(T - B)/E$. E specifies that only one of every E draws are saved. $E > 1$ reduces autocorrelation between obtained samples at the expense of computation time.

`lml` Logical indicating if the log-marginal likelihood should be approximated. Default is `FALSE`, which reduces computation time. Log-marginal likelihood is approximated using methods in (Chib 1995).

`ybal` Logical indicating if the mass balanced samples for each y should be returned. Default is `TRUE`. Setting `ybal=FALSE` results in a savings in RAM and computation time.

`verb` Numeric indicating verbosity of progress printed to R-console. The default of 1 prints messages and a progress bar to the console during all iterative methods. `verb = 0` indicates no messages are printed.

Details

See `vignette("Two_Node_Process",package = "BayesMassBal")` for further details on how to use function outputs.

When the `priors` argument is left unspecified, a set of default conjugate priors are used, which are chosen to provide little information to the posterior distribution when adequate data is supplied. In the current version of the `BayesMassBal` package, only the conjugate priors stated below can be used, but hyperparameter values can be specified by the user.

The prior distribution on `beta` is a normal distribution truncated at 0. The mean of this distribution, before truncation, is 0 for each element of `beta`. Each `beta` is independent with a variance of 10,000,000 before truncation.

When `cov.structure = "indep"` all observations in a sample set are independent. A Gamma prior distribution, with $a = 1$ and $b = 100,000,000$, is placed on the precision of the mass flow rate for each sample component at each sample location. Parameterization used for the Gamma distribution is:

$$f(\phi) = \frac{b^a}{\Gamma(a)} \phi^{a-1} e^{-b\phi}$$

When `cov.structure = "component"` or `"location"`, the prior distribution on Σ_i is inverse Wishart$(\nu_0, \nu_0 \times S_0)$. The degrees of freedom parameter, ν_0, is equal to the dimension of Σ_i. The scale matrix parameter is equal to a matrix, S_0, with the sample variance of the observation being correlated on the diagonal, multiplied by ν_0.

The user is able to specify the prior hyperparameters of the mean and variance of `beta`, a and b for each ϕ, and the degrees of freedom and scale matrix for each Σ_i.

Value

Returns a list of outputs

`beta`	List of matrices of samples from the distribution of reconciled data. Each matrix in the list is a separate sample component. Each column of a matrix in `beta` is a draw from the target distribution.
`Sig`	List of matrices containing draws from the distribution of each covariance matrix. If `S.t` is the t^{th} draw from the distribution of covariance matrix `S` and: • `cov.structure = "indep"`, the t^{th} column of a matrix in `Sig` is `diag(S.t)`. • `cov.structure = "component"` or `"location"`, the t^{th} column of a matrix in `Sig` is equal to `S.t[upper.tri(W,diag = TRUE)]`.
`priors`	List of prior hyperparameters used in generating conditional posterior distributions and approximating log-marginal likelihood. The structure of the input argument `priors` is required to be the same as the structure of this returned list slice. See Details.
`cov.structure`	Character string containing the covariance structure used.
`y.cov`	List of character matrices indicating details for the structure of each covariance matrix. Only returned when `cov.structure = "location"`
`lml`	Numeric of the log-marginal likelihood approximation. Returns `NA` when `lml = FALSE`
`ybal`	List of samples from the distribution of reconciled mass flow rates, in the same format as the function argument `y`. Produced with argument `ybal = TRUE`. Equivalent to `lapply(BMB(...)$beta,function(X,x){x %*% X} ,x = X)` . Viewing this output is more intuitive than viewing samples of `beta`, at the expense of RAM and some computation time.
`X`	The function argument `X` is passed to the output so that it can be used with other `BayesMassBal` functions.

References

Chib S (1995). "Marginal likelihood from the Gibbs output." *Journal of the american statistical association*, **90**(432), 1313–1321. Casella G, George EI (1992). "Explaining the Gibbs sampler." *The American Statistician*, **46**(3), 167–174.

Examples

```
y <- importObservations(file = system.file("extdata", "twonode_example.csv",
                                    package = "BayesMassBal"),
                    header = TRUE, csv.params = list(sep = ";"))

C <- matrix(c(1,-1,0,-1,0,0,1,-1,0,-1), byrow = TRUE, ncol = 5, nrow = 2)
X <- constrainProcess(C = C)

BMB_example <- BMB(X = X, y = y, cov.structure = "indep",
                   BTE = c(10,300,1), lml = FALSE, verb=0)
```

`constrainProcess` *Matrix Constraining Process*

Description

Generates matrix X which maps constrained masses β to observed masses y for an individual sample component, when given a linear system of constraining equations.

Usage

```
constrainProcess(C = NULL, file = FALSE)
```

Arguments

`C`	Matrix of constraints for a process at steady state. See Details below.
`file`	Character string indicating file path for a `*.csv` file containing linear constraints. Only values of -1, 0, and 1 are valid. The first row in the `file` is required to be a header naming the sampling locations.

Details

The output of this function is meant to be used as the input parameter `X` in the `BMB` function. The matrix `C`, or imported matrix from `file`, indexes sampling locations via columns, and number of constraints via rows. Only values of -1, 0, and 1 are valid, and indicate mass leaving a node, a location that is not relevant to a node, and mass entering a node respectively. Constraints should only be indicated around each node. No additional, and no less constraints should be specified. Additional constraints are redundant. Each sample component is subject to the same constraints, and therefore the constraints given to `constrainProcess` do not need to be repeated for each component.

Value

Returns the matrix X which maps β to observed masses y. No changes need to be made to `X` when using with `BMB`.

Examples

```
## For a single node process where
## y_1 = y_2 + y_3

C <- matrix(c(1,-1,-1), nrow=  1,ncol = 3)
constrainProcess(C = C)

## For a 2 node process with 1 input and 3 outputs
## as shown in \dontrun{vignette("Two_Node_Process", package = "BayesMassBal")}

C <- matrix(c(1,-1,0,-1,0,0,1,-1,0,-1), byrow = TRUE, ncol = 5, nrow = 2)
constrainProcess(C = C)
```

`importObservations` *Import Observed Mass Flow Rates*

Description

Imports observed mass flow rates stored in a `*.csv` file and then organizes the data for use with the `BMB` function.

Usage

```
importObservations(file, header = FALSE, csv.params = NULL)
```

Arguments

`file`	Character string containing the name of `*.csv` from file which data will be read. See Details below for valid file structures.
`header`	Logical indicating if the first row of `file` file contains header information. Current implementation of `importObservations` discards this information.
`csv.params`	List of arguments to be passed to `read.csv`

Details

The purpose of this function is to make it easy to import and structure loosely organized data contained in a `*.csv` into a list for use as the `y` argument passed to the `BMB` function. The entries in file must be organized as such:

- The first column of `file` must contain an integer sample location. The value of this integer must correspond to the column number used to specify linear constraints in `constrainProcess`. For example, data for a given component collected at sampling location y_2 should be indicated with a `2` in the first column of `file` used with `importObservations`. In the `file` used with `constrainProcess`, the linear constraint(s) on y_2 are indicated in the second column.
- The second column of `file` must contain sample component names. **This field is case sensitive**. Ensure a given sample component is named consistently, including capitalization and spacing.
- Columns 3 to $K+2$ of `file` must contain observed mass flow rates for the K collected sample sets. All observations located in the same column should be collected at the same time.
- Sample components of interest must be specified for each location. If a sample component is not detected at some locations, but is detected at others, this component should be included in `file` with a specified mass flow rate of 0, or a very small number.

`importObservations` reads the contents of `file`, sorts the sampling locations numerically, then creates a list of data frames. Each data frame contains the data for a single sample component.

Value

Returns a list of data frames. Each data frame is named according to the unique sample components specified in the second column of `file`. This list object is intended to be used as the argument `y` for the `BMB` function.

Examples

```
 y <- importObservations(file = system.file("extdata", "twonode_example.csv",
                                     package = "BayesMassBal"),
                    header = TRUE, csv.params = list(sep = ";"))

## The linear constraints for this example data set are:
C <- matrix(c(1,-1,0,-1,0,0,1,-1,0,-1), byrow = TRUE, ncol = 5, nrow = 2)

## The X matrix for this data set can be found using:
X <- constrainProcess(C = C)
```

`mainEff` *Main Effects*

Description

Calculates the main effect of a variable, which is independent of process performance, on a function.

Usage

```
mainEff(
  BMBobj,
  fn,
  rangex,
  xj,
  N = 50,
  res = 100,
  hdi.params = c(1, 0.95),
  ...
)
```

Arguments

`BMBobj`	A `BayesMassBal` object originally obtained from the `BMB` function. See `BMB`.
`fn`	A character string naming a function with arguments of `BMBobj$ybal` and independent random variables `X`. See Details and examples for more on function requirements.
`rangex`	A numeric matrix. Each column of `rangex` contains the minimum and maximum value of uniformly distributed random values making up vector x.
`xj`	Integer indexing which element in x is used for conditioning for $E_x[f(x, y)\|x_j]$. If a vector is supplied the marginal main effect of each element is calculated sequentially. The integers supplied in `xj` are equivalent to the indices of the columns in `rangex`.
`N`	Integer specifying the length of the sequence used for `xj`. Larger `N` trades a higher resolution of the main effect of `xj` for longer computation time and larger RAM requirements.

res	Integer indicating the number of points to be used for each Monte-Carlo integration step. Larger res reduces Monte-Carlo variance as the expense of computation time.
hdi.params	Numeric vector of length two, used to calculate Highest Posterior Density Interval (HPDI) of the main effect xj using hdi. hdi.params[1] = 1 indicates hdi is used, and the mean and HPDI bounds are returned instead of the every sample from the distribution of $E_x[f(x,y)\|x_j]$. The second element of hdi is passed to the credMass argument in the hdi function. The default, hdi.params = c(1,0.95), returns 95% HPDI bounds.
...	Extra arguments passed to the named fn

Details

The mainEff function returns a distribution of $E_x[f(x,y)|x_j]$, marginalized over the samples of BMBobj$ybal, giving the distribution of $E_x[f(x,y)|x_j]$ which incorporates uncertainty of a chemical or particulate process.

In the current implementation of mainEff in the BayesMassBal package, only uniformly distributed values of x are supported.

The $f(x,y)$ is equivalent to the supplied function named in mainEff(fn). For the arguments of fn, ybal is structured in a similar manner as BMBobj$ybal. The only difference being individual columns of each matrix are used at a time, and are vectorized. Note the way ybal is subset in the example function fn_example. The supplied X is a matrix, with columns corresponding to each element in x. The output to fn must be a vector of length nrow(x). The first argument of fn must be X, the second argument must be BMBobj$ybal. Order of other arguments passed to fn through ... does not matter. Look at the example closely for details!

Value

A list of length(xj) list(s). Each list specifies output for the main effect of a xj

g	The grid used for a particular xj
fn.out	A matrix giving results on $E_x[f(x,y)\|x_j]$. If hdi.params[1] = 1, the mean and Highest Posterior Density Interval (HPDI) bounds of $E_x[f(x,y)\|x_j]$ are returned. Otherwise, samples of $E_x[f(x,y)\|x_j]$ are returned. The index of each column of fn.out corresponds to the the value of g at the same index.
fn	Character string giving the name of the function used. Same value as argument fn.
xj	Integer indicating the index of x corresponding to a grouped fn.out and g.

Examples

```
## Importing Data, generating BMB object
y <- importObservations(file = system.file("extdata", "twonode_example.csv",
                                    package = "BayesMassBal"),
                  header = TRUE, csv.params = list(sep = ";"))

C <- matrix(c(1,-1,0,-1,0,0,1,-1,0,-1), byrow = TRUE, ncol = 5, nrow = 2)
```

```
X <- constrainProcess(C = C)

BMB_example <- BMB(X = X, y = y, cov.structure = "indep",
                   BTE = c(10,200,1), lml = FALSE, verb=0)

fn_example <- function(X,ybal){
    cu.frac <- 63.546/183.5
    feed.mass <- ybal$CuFeS2[1] + ybal$gangue[1]
    ## Concentrate mass per ton feed
    con.mass <- (ybal$CuFeS2[3] + ybal$gangue[3])/feed.mass
    ## Copper mass per ton feed
    cu.mass <- (ybal$CuFeS2[3]*cu.frac)/feed.mass
    gam <- c(-1,-1/feed.mass,cu.mass,-con.mass,-cu.mass,-con.mass)
    f <- X %*% gam
    return(f)
    }

rangex <- matrix(c(4.00 ,6.25,1125,1875,3880,9080,20,60,96,208,20.0,62.5),
                  ncol = 6, nrow = 2)

mE_example <- mainEff(BMB_example, fn = "fn_example",rangex =  rangex,xj = 3, N = 15, res = 4)
```

`plot.BayesMassBal` *Plots BayesMassBal Object*

Description

Visualizes data from a `BayesMassBal` class object in a user specified way. Options include trace plots, posterior densities, and main effects plots. Meant to be a quick diagnostic tool, and not to produce publication quality plots.

Usage

```
## S3 method for class 'BayesMassBal'
plot(
  x,
  sample.params = NA,
  layout = c("trace", "dens"),
  hdi.params = c(1, 0.95),
  ...
)
```

Arguments

`x`	A `BayesMassBal` object returned from the `BMB` function
`sample.params`	List to be used for indicating model parameter samples used for creation of plot(s). See details for required structure.

layout	Character string indicating the desired data to be plotted. "trace" produces trace plots of sequential parameter draws. "dens" produces densities of posterior draws.
hdi.params	Numeric vector of length two, used to draw Highest Posterior Density Intervals (HPDI) using hdi, and otherwise ignored. hdi.params[1] = 1 indicates hdi bounds should be drawn. The second element of hdi is passed to credMass in the hdi function. The default, hdi.params = c(1,0.95), plots the 95% HPDI bounds.
...	Passes extra arguments to plot()

Details

The list of params requires a specific structure dependent on the choice of layout and the desired plots.

If layout = "trace" or layout = "dens", names(list) must contain each model parameter desired for plotting. The structure under the model parameter names must be the same as to the structure of the relevant subset of the BayesMassBal object to be used. For example, if a BayesMassBal object is created using a process with sample components c("CuFeS2","gangue") and the users wants plots of reconciled masses y_1 and y_2 for both components to be created, params = list(y.bal = list(CuFeS2 = c(1,2),gangue = c(1,2)) should be used. Note, str(params) mimics str(x), while the vectors listed simply index the desired model parameters to be plotted.

See vignette("Two_Node_Process",package = "BayesMassBal") for an example of the required structure.

Value

Plots BayesMassBal object based on arguments passed to plot.

---

twonodeSim *Two Node Process Data Simulation*

Description

Simulates data for a stochastic two node, two component process at steady state. Location indices are the same as what is shown in vignette("Two_Node_Process",package = "BayesMassBal").

Usage

```
twonodeSim(
  K = 7,
  feed = list(rate = 100, sd = 6, CuFeS2grade = 1.2),
  rec = list(CuFeS2 = list(mean = c(98, 95)/100, var = c(5e-05, 8e-05)), gangue =
    list(mean = c(7, 4)/100, var = c(5e-05, 2.5e-05))),
 assayNoise = list(CuFeS2 = c(0.15, 0.2, 0.05, 5e-05, 0.005), gangue = c(5, 1, 0.03,
    2, 0.5)),
  truncation = TRUE
)
```

Arguments

K	Numeric specifying the number of sample sets to be simulated.
feed	List specifying qualities for the process grade. See default for required structure. rate is the mean feed rate. sd is the standard deviation of the feed rate. CuFeS2grade is the mass percent CuFeS2 present in the feed. Grade is not stochastic. See Details for important information on specifying these values.
rec	List specifying mean and variance in process performance. See default for required structure. rec$component$mean is a vector giving mean fractional recovery of the given component for c(node1,node2). rec$component$var gives the variance in the process in a similar manner. See Details.
assayNoise	List specifying standard deviations of random noise added to simulated process outputs. See default for required structure. The index of a vector within the list is equivalent to the index of the sampling location. See Details section for important information on specifying these values.
truncation	Logical indicating if the simulation should be rerun, and previous results discarded, until no observed values are less than 0. Default is TRUE. See details for more information.

Details

Each of the K data sets collected from the twonodeSim() simulation is independent and identically distributed.

The feed rate to the process is normally distributed with a mean of feed$rate, and a standard deviation of feed$sd. If the feed rate is sufficiently small, and the standard deviation is sufficiently large, negative feed rates can be generated.

Process recovery at each node is simulated from a beta distribution, reparameterized as shown in this post to make parameter specification more intuitive. This reparameterization is only valid when $\sigma^2 \leq \mu(1-\mu)$, and the list argument rec must be specified as such.

The steps of the simulation for each sample set are:

1. Draw a random normally distributed feed rate.
2. Draw random values for recovery of the two components at each node.
3. Calculate mass flow rate at each location. These mass flow rates are the *true* mass flow rates, given the process variability.
4. Adds normally distributed noise to each observation as specified in argument assayNoise

If the standard deviations supplied to feed and assayNoise are sufficiently large, the simulation can return negative mass flow rates.

The argument truncation = TRUE discards negative mass flow rates, and reruns the simulation until all values are non-negative. For some combinations of a large K and specifications in feed and assayNoise, this can happen frequently. If if the simulation is run three or more times a warning will be printed that the returned expectations are unreliable. If this is the case, expectations should be calculated using analytical or Monte-Carlo methods outside of the abilities of this function. For the default parameters, truncation can occur, but is rare. The default parameters were chosen in a way that makes a truncation warning highly unlikely.

Value

Returns a list of simulated data and expected values. List members are as follows:

`simulation`	List of matrices giving simulated data. `twonodeSim()$simulation` is structured so that it can directly be passed to the BMB function as the `y` argument.
`expectations`	List of matrices giving expected values of the mass flow rate for each component at every location. See the Details section for information about instances that may create reliability issues with this output.

Examples

```
y <- twonodeSim()$simulation

## Then the BMB function can be run as
C <- matrix(c(1,-1,0,-1,0,0,1,-1,0,-1), byrow = TRUE, ncol = 5, nrow = 2)
X <- constrainProcess(C = C)

BMB(X = X, y = y, BTE = c(100,600,1))
```

Index