Package 'bhpm'

April 1, 2020

Type Package

Title Bayesian Hierarchical Poisson Models for Multiple Grouped Outcomes with Clustering

Version 1.7

Date 2020-03-31

Author Raymond Carragher [aut, cre] (<https://orcid.org/0000-0002-0120-625X>)

Maintainer Raymond Carragher <raymond.carragher@strath.ac.uk>

Depends coda

Encoding UTF-8

Description Bayesian hierarchical methods for the detection of differences in rates of related outcomes for multiple treatments for clustered observations. Theoretical background for the models is given in Carragher (2017) <https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.736866>. The models in this package are extensions for multiple treatments and clusters. This software was developed for the Precision Drug Theraputics: Risk Prediction in Pharmacoepidemiology project as part of a Rutherford Fund Fellowship at Health Data Research (UK), Medical Research Council (UK) award reference MR/S003967/1 (<https://gtr.ukri.org/>). Principal Investigator: Raymond Carragher.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-04-01 09:50:18 UTC

R topics documented:

`bhpm-package` *Bayesian Hierarchical Possion Models for Mulitple Grouped Outcomes with Clustering*

Description

This package implements Bayesian Hierarchical models for the detection of differences in rates of outcomes for multiple treatments for clustered observations using groupings of outcomes.

Details

The methods implemented use assumed groupings of outcomes to detect differences in the occurrence of the outcome for different treatments. Methods based on Bayesian Hierarchical models are provided. The methods are extension of those described in Carragher (2017) for multiple treatments and clusters.

Author(s)

R. Carragher<raymond.carragher@strath.ac.uk; rcarragh@gmail.com>

References

Carragher R. Detection of Safety Signals in Randomised Controlled Trials. PhD thesis. University of Strathclyde, 2017 <https://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.736866>.

bhpm.cluster.1a.hier2 *A Two-Level Hierarchical Model for Grouped Data with Clusters and without Point-Mass.*

Description

Implementation of a Two-Level Hierarchical for Grouped Data with Clusters and without Point-Mass.

Usage

```
bhpm.cluster.1a.hier2(cluster.data, sim_type = "SLICE", burnin = 10000,
iter = 40000, nchains = 3,
global.sim.params = data.frame(type = c("MH", "SLICE"),
param = c("sigma_MH", "w"), value = c(0.2,1), control = c(0,6),
stringsAsFactors = FALSE),
sim.params = NULL,
monitor = data.frame(variable = c("theta", "gamma", "mu.gamma",
"mu.theta", "sigma2.theta", "sigma2.gamma"),
monitor = c(1, 1, 1, 1, 1, 1),
stringsAsFactors = FALSE),
initial_values = NULL,
level = 1,
hyper_params = list(mu.gamma.0 = 0, tau2.gamma.0 = 10, mu.theta.0 = 0,
tau2.theta.0 = 10, alpha.gamma = 3, beta.gamma = 1,
alpha.theta = 3, beta.theta = 1),
memory_model = "HIGH")
```

Arguments

cluster.data	A file or data frame containing the cluster data. It must contain the columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp* (1 - control, 2,... comparator treatments), *Count* (total number of events), *Exposure* (total exposure of time of all patients for the Trt.Grp in the Cluster).
sim_type	The type of MCMC method to use for simulating from non-standard distributions. Allowed values are *"MH"* and *"SLICE"* for Metropis_Hastings and Slice sampling respectively.
burnin	The burnin period for the monte-carlo simulation. These are discarded from the returned samples.
iter	The total number of iterations for which the monte-carlo simulation is run. This includes the burnin period. The total number of samples returned is *iter - burnin*
nchains	The number of independent chains to run.
global.sim.params	A data frame containing the parameters for the simuation type *sim_type*. For *"MH"* the parameter is the variance of the normal distribution used to simulate

the next candidate value centred on the current value. For *"SLICE"* the parameters are the estimated width of the slice and a value limiting the search for the next sample.

`sim.params` A dataframe containing simulation parameters which override the global simulation parameters (*global.sim.params*) for particular model parameters. *sim.params* must contain the following columns: type: the simulation type ("MH" or "SLICE"); variable: the model parameter for which the simulation parameters are being overridden; Outcome.Grp: the outcome grouping (if applicable); Outcome: the outcome (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

The function *bhpm.sim.control.params* generates a template for *sim.params* which can be edited by the user.

`monitor` A dataframe indicating which sets of variables to monitor.

`initial_values` The initial values for starting the chains. If NULL (the default) is passed the function generates the initial values for the chains. initial_values is a list with the following format:

```
list(gamma, theta, mu.gamma, mu.theta, sigma2.gamma,
sigma2.theta)
```

where each element of the list is either a dataframe or array. The function *bhpm.gen.initial.values* can be used to generate a template for the list which can be updated by the user if required.

`level` The level of dependancy between the clusters. 0 - independent clusters, 1 - common cluster means. dependancy.

`hyper_params` The hyperparameters for the model.

`memory_model` Allowed values are "HIGH" and "LOW". "HIGH" means use as much memory as possible. "LOW" means use the minimum amount of memory.

Details

The model is fitted by a Gibbs sampler. The posterior distributions for *gamma* and *theta* are sampled with either a Metropolis-Hastings step or a slice sampler.

Value

The output from the simulation including all the sampled values is as follows:

```
list(id, sim_type, chains, nClusters, Clusters, Trt.Grps, nOutcome.Grp, maxOutcome.Grps,
maxOutcomes, nOutcome, Outcome, Outcome.Grp, burnin, iter, monitor,
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, gamma,
theta, gamma_acc, theta_acc)
```

where

id - a string identifying the verion of the function

sim_type - an string identifying the samlping method used for non-standard distributions, either *"MH"* or *"SLICE"*

chains - the number of chains for which the simulation was run.

nClusters - the number of clusters in the simulation

Clusters - an array. The clusters.

Trt.Grps - an array. The treatments.

nOutcome.Grp - the number of outcome groupings.

maxOutcome.Grps - the maximum number of outcome groupings in a cluster.

maxOutcomes - the maximum number of outcomes in an outcome grouping.

nOutcome - an array. The number of outcomes in each outcome grouping.

Outcome - an array of dimension *nOutcome.Grp*, *nOutcome*. The outcomes.

Outcome.Grp - an array. The outcome groupings.

burnin - burn-in used for the simulation.

iter - the total number of iterations in the simulation.

monitor - the variables being monitored. A dataframe.

mu.gamma - array of generated samples.

mu.theta - array of generated samples.

sigma2.gamma - array of generated samples.

sigma2.theta - array of generated samples.

gamma - array of generated samples.

theta - array of generated samples.

gamma_acc - array of generated samples.

theta_acc - the acceptance rate for the theta samples if a Metropolis-Hastings method is used.

Note

The function performs the simulation and returns the raw output. No checks for convergence are performed.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
raw = bhpm.cluster.1a.hier2(bhpm.cluster.data1, level = 1, burnin = 100, iter = 200)

data(bhpm.cluster.data1)
raw = bhpm.cluster.1a.hier2(bhpm.cluster.data1, level = 1)
```

bhpm.cluster.1a.hier3 *A Three-Level Hierarchical Model for Grouped Data with Clusters and without Point-Mass.*

Description

Implementation of a Three-Level Hierarchical for Grouped Data with Clusters and without Point-Mass.

Usage

```
bhpm.cluster.1a.hier3(cluster.data, sim_type = "SLICE", burnin = 10000,
iter = 40000, nchains = 3,
global.sim.params = data.frame(type = c("MH", "SLICE"),
param = c("sigma_MH", "w"), value = c(0.2,1), control = c(0,6),
stringsAsFactors = FALSE),
sim.params = NULL,
monitor = data.frame(variable = c("theta", "gamma", "mu.gamma",
"mu.theta", "sigma2.theta", "sigma2.gamma",
"mu.theta.0", "mu.gamma.0", "tau2.theta.0", "tau2.gamma.0"),
monitor = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
stringsAsFactors = FALSE),
initial_values = NULL,
level = 1,
hyper_params = list(mu.gamma.0.0 = 0, tau2.gamma.0.0 = 10,
mu.theta.0.0 = 0, tau2.theta.0.0 = 10, alpha.gamma.0.0 = 3,
beta.gamma.0.0 = 1, alpha.theta.0.0 = 3, beta.theta.0.0 = 1,
alpha.gamma = 3, beta.gamma = 1,
alpha.theta = 3, beta.theta = 1),
memory_model = "HIGH")
```

Arguments

cluster.data	A file or data frame containing the cluster data. It must contain the columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp* (1 - control, 2,... comparator treatments), *Count* (total number of events), *Exposure* (total exposure of time of all patients for the Trt.Grp in the Cluster).
sim_type	The type of MCMC method to use for simulating from non-standard distributions. Allowed values are *"MH"* and *"SLICE"* for Metropis_Hastings and Slice sampling respectively.
burnin	The burnin period for the monte-carlo simulation. These are discarded from the returned samples.
iter	The total number of iterations for which the monte-carlo simulation is run. This includes the burnin period. The total number of samples returned is *iter - burnin*
nchains	The number of independent chains to run.

`global.sim.params`
: A data frame containing the parameters for the simuation type *sim_type*. For *"MH"* the parameter is the variance of the normal distribution used to simulate the next candidate value centred on the current value. For *"SLICE"* the parameters are the estimated width of the slice and a value limiting the search for the next sample.

`sim.params`
: A dataframe containing simulation parameters which override the global simulation parameters (*global.sim.params*) for particular model parameters. *sim.params* must contain the following columns: type: the simulation type ("MH" or "SLICE"); variable: the model parameter for which the simulation parameters are being overridden; Outcome.Grp (if applicable); Outcome (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

  The function *bhpm.sim.control.params* generates a template for *sim.params* which can be edited by the user.

`monitor`
: A dataframe indicating which sets of variables to monitor.

`initial_values`
: The initial values for starting the chains. If NULL (the default) is passed the function generates the initial values for the chains. initial_values is a list with the following format:

  ```
  list(gamma, theta, mu.gamma, mu.theta, sigma2.gamma,
  sigma2.theta, mu.gamma.0, mu.theta.0, tau2.gamma.0,
  tau2.theta.0)
  ```

  where each element of the list is either a dataframe or array. The function *bhpm.gen.initial.values* can be used to generate a template for the list which can be updated by the user if required.

`level`
: The level of dependancy between the clusters. 0 - independent clusters, 1 - common cluster means, 2 - weak dependancy between clusters.

`hyper_params`
: The hyperparameters for the model.

`memory_model`
: Allowed values are "HIGH" and "LOW". "HIGH" means use as much memory as possible. "LOW" means use the minimum amount of memory.

Details

The model is fitted by a Gibbs sampler. The posterior distributions for *gamma* and *theta* are sampled with either a Metropolis-Hastings step or a slice sampler.

Value

The output from the simulation including all the sampled values is as follows:

```
list(id, sim_type, chains, nClusters, Clusters, nOutcome.Grp, maxOutcome.Grps,
maxOutcomes, nOutcome, Outcome, Outcome.Grp, burnin, iter, monitor,
mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0,
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, gamma,
theta, gamma_acc, theta_acc)
```

where

id - a string identifying the verion of the function

sim_type - an string identifying the samlping method used for non-standard distributions, either *"MH"* or *"SLICE"*

chains - the number of chains for which the simulation was run.

nClusters - the number of clusters in the simulation

Clusters - an array. The clusters.

nOutcome.Grp - the number of outcome groupings.

maxOutcome.Grps - the maximum number of outcome groupings in a cluster.

maxOutcomes - the maximum number of outcomes in an outcome grouping.

nOutcome - an array. The number of outcomes in each outcome grouping.

Outcome - an array of dimension *nOutcome.Grp*, *maxOutcomes*. The outcomes.

Outcome.Grp - an array. The outcome groupings.

burnin - burnin used for the simulation.

iter - the total number of iterations in the simulation.

monitor - the variables being monitored. A dataframe.

mu.gamma.0 - array of generated samples.

mu.theta.0 - array of generated samples.

tau2.gamma.0 - array of generated samples.

tau2.theta.0 - array of generated samples.

mu.gamma - array of generated samples.

mu.theta - array of generated samples.

sigma2.gamma - array of generated samples.

sigma2.theta - array of generated samples.

gamma - array of generated samples.

theta - array of generated samples.

gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used.

theta_acc - the acceptance rate for the theta samples if a Metropolis-Hastings method is used.

Note

The function performs the simulation and returns the raw output. No checks for convergence are performed.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
raw = bhpm.cluster.1a.hier3(bhpm.cluster.data1, level = 1, burnin = 100, iter = 200)

data(bhpm.cluster.data1)
raw = bhpm.cluster.1a.hier3(bhpm.cluster.data1, level = 1)
```

---

bhpm.cluster.BB.hier2 *A Two-Level Hierarchical Model for grouped data and clusters with Point-Mass.*

Description

Implementation of a Two-Level Hierarchical for grouped data and clusters with Point-Mass.

Usage

```
bhpm.cluster.BB.hier2(cluster.data, sim_type = "SLICE", burnin = 20000,
iter = 60000, nchains = 5, theta_algorithm = "MH",
global.sim.params = data.frame(type = c("MH", "MH", "MH", "MH",
"SLICE", "SLICE", "SLICE"),
param = c("sigma_MH_alpha", "sigma_MH_beta", "sigma_MH_gamma",
"sigma_MH_theta", "w_alpha", "w_beta", "w_gamma"),
value = c(3, 3, 0.2, 0.5, 1, 1, 1), control = c(0, 0, 0, 0, 6, 6, 6),
stringsAsFactors = FALSE),
sim.params = NULL,
monitor = data.frame(variable = c("theta", "gamma", "mu.gamma",
"mu.theta", "sigma2.theta", "sigma2.gamma", "pi"),
monitor = c(1, 1, 1, 1, 1, 1, 1), stringsAsFactors = FALSE),
initial_values = NULL, level = 1,
hyper_params = list(mu.gamma.0 = 0, tau2.gamma.0 = 10, mu.theta.0 = 0,
tau2.theta.0 = 10, alpha.gamma = 3, beta.gamma = 1, alpha.theta = 3,
beta.theta = 1, alpha.pi = 1.1, beta.pi = 1.1),
global.pm.weight = 0.5,
pm.weights = NULL,
adapt_phase=1, memory_model = "HIGH")
```

Arguments

cluster.data	A file or data frame containing the cluster data. It must contain the columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp* (1 - control, 2,... comparator treatments), *Count* (total number of events), *Exposure* (total exposure of time of all patients fot the Trt.Grp in the Cluster).
burnin	The burnin period for the monte-carlo simulation. These are discarded from the returned samples.

`iter` The total number of iterations for which the monte-carlo simulation is run. This includes the burnin period. The total number of samples returned is *iter - burnin*

`nchains` The number of independent chains to run.

`theta_algorithm`
MCMC algorithm used to sample the theta variables. "MH" is the only currently supported stable algorithm.

`sim_type` The type of MCMC method to use for simulating from non-standard distributions apart from theta. Allowed values are *"MH"* and *"SLICE"* for Metropis_Hastings and Slice sampling respectively.

`monitor` A dataframe indicating which sets of variables to monitor.

`global.sim.params`
A data frame containing the parameters for the simuation type *sim_type*. For *"MH"* the parameter is the variance of the normal distribution used to simulate the next candidate value centred on the current value. For *"SLICE"* the parameters are the estimated width of the slice and a value limiting the search for the next sample.

`sim.params` A dataframe containing simulation parameters which override the global simulation parameters (*global.sim.params*) for particular model parameters. *sim.params* must contain the following columns: type: the simulation type ("MH" or "SLICE"); variable: the model parameter for which the simulation parameters are being overridden; Outcome.Grp: the outcome grouping (if applicable); Outcome: the outcome (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

The function *bhpm.sim.control.params* generates a template for *sim.params* which can be edited by the user.

`initial_values` The initial values for starting the chains. If NULL (the default) is passed the function generates the initial values for the chains. initial_values is a list with the following format:

```
list(gamma, theta, mu.gamma, mu.theta, sigma2.gamma,
sigma2.theta, pi, mu.gamma.0, mu.theta.0,
tau2.gamma.0, tau2.theta.0, alpha.pi, beta.pi)
```

The function *bhpm.gen.initial.values* can be used to generate a template for the list which can be updated by the user if required.

`level` Allowed valus are 0, 1. Respectively these indicate independent clusters, common means across the clusters.

`hyper_params` The hyperparameters for the model.

`global.pm.weight`
A global weighting for the proposal distribution used to sample theta.

`pm.weights` Override global.pm.weight for specific outcomes.

`adapt_phase` Unused parameter.

`memory_model` Allowed values are "HIGH" and "LOW". "HIGH" means use as much memory as possible. "LOW" means use the minimum amount of memory.

Details

The model is fitted by a Gibbs sampler.

Value

The output from the simulation including all the sampled values is as follows:

```
list(id, theta_alg, sim_type, chains, nClusters, Clusters, Trt.Grps, nOutcome.Grp,
maxOutcome.Grps, maxOutcomes, nOutcome, Outcome, Outcome.Grp, burnin,
iter, monitor,
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi,
gamma, theta, gamma_acc, theta_acc)
```

where

id - a string identifying the verions of the function

theta_alg - an string identifying the algorithm used to smaple theta

sim_type - an string identifying the samlping method used for non-standard distributions, either *"MH"* or *"SLICE"*

chains - the number of chains for which the simulation was run

nClusters - the number of clusters in the simulation

Clusters - an array. The clusters.

nOutcome.Grp - the number of outcome groupings.

maxOutcome.Grps - the maximum number of outcome groupings in a cluster.

maxOutcomes - the maximum number of outcomes in an outcome grouping.

nOutcome - an array. The number of outcomes in each outcome grouping.

Outcome - an array of dimension *nOutcome.Grp*, *maxOutcomes*. The outcomes.

Outcome.Grp - an array. The outcome groupings.

burnin - burnin used for the simulation.

iter - the total number of iterations in the simulation.

monitor - the variables being monitored. A dataframe.

mu.gamma - array of generated samples.

mu.theta - array of generated samples.

sigma2.gamma - array of generated samples.

sigma2.theta - array of generated samples.

pi - array of generated samples.

gamma - array of generated samples.

theta - array of generated samples.

gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used.

theta_acc - the acceptance rate for the theta samples.

Note

The function performs the simulation and returns the raw output. No checks for convergence are performed.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
raw = bhpm.cluster.BB.hier2(bhpm.cluster.data1, level = 1, burnin = 100, iter = 200)

data(bhpm.cluster.data1)
raw = bhpm.cluster.BB.hier2(bhpm.cluster.data1, level = 1)
```

---

bhpm.cluster.BB.hier3 *A Three-Level Hierarchical Model for grouped data and clusters with Point-Mass.*

Description

Implementation of a Three-Level Hierarchical for grouped data and clusters with Point-Mass.

Usage

```
bhpm.cluster.BB.hier3(cluster.data, sim_type = "SLICE", burnin = 20000,
iter = 60000, nchains = 5, theta_algorithm = "MH",
global.sim.params = data.frame(type = c("MH", "MH", "MH", "MH",
"SLICE", "SLICE", "SLICE"),
param = c("sigma_MH_alpha", "sigma_MH_beta", "sigma_MH_gamma",
"sigma_MH_theta", "w_alpha", "w_beta", "w_gamma"),
value = c(3, 3, 0.2, 0.25, 1, 1, 1), control = c(0, 0, 0, 0, 6, 6, 6),
stringsAsFactors = FALSE),
sim.params = NULL,
monitor = data.frame(variable = c("theta", "gamma", "mu.gamma", "mu.theta",
"sigma2.theta", "sigma2.gamma",
"mu.theta.0", "mu.gamma.0", "tau2.theta.0", "tau2.gamma.0",
"pi", "alpha.pi", "beta.pi"),
monitor = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1),
stringsAsFactors = FALSE),
initial_values = NULL, level = 1, hyper_params = list(mu.gamma.0.0 = 0,
tau2.gamma.0.0 = 10, mu.theta.0.0 = 0, tau2.theta.0.0 = 10,
alpha.gamma.0.0 = 3, beta.gamma.0.0 = 1, alpha.theta.0.0 = 3,
beta.theta.0.0 = 1, alpha.gamma = 3,
beta.gamma = 1, alpha.theta = 3, beta.theta = 1,
```

```
lambda.alpha = 1.0, lambda.beta = 1.0),
global.pm.weight = 0.5,
pm.weights = NULL,
adapt_phase=1, memory_model = "HIGH")
```

Arguments

`cluster.data`	A file or data frame containing the cluster data. It must contain the columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp* (1 - control, 2,... comparator treatments), *Count* (total number of events), *Exposure* (total exposure of time of all patients fot the Trt.Grp in the Cluster).
`burnin`	The burnin period for the monte-carlo simulation. These are discarded from the returned samples.
`iter`	The total number of iterations for which the monte-carlo simulation is run. This includes the burnin period. The total number of samples returned is *iter - burnin*
`nchains`	The number of independent chains to run.
`theta_algorithm`	MCMC algorithm used to sample the theta variables. "MH" is the only currently supported stable algorithm.
`sim_type`	The type of MCMC method to use for simulating from non-standard distributions apart from theta. Allowed values are *"MH"* and *"SLICE"* for Metropis_Hastings and Slice sampling respectively.
`monitor`	A dataframe indicating which sets of variables to monitor.
`global.sim.params`	A data frame containing the parameters for the simuation type *sim_type*. For *"MH"* the parameter is the variance of the normal distribution used to simulate the next candidate value centred on the current value. For *"SLICE"* the parameters are the estimated width of the slice and a value limiting the search for the next sample.
`sim.params`	A dataframe containing simulation parameters which override the global simulation parameters (*global.sim.params*) for particular model parameters. *sim.params* must contain the following columns: type: the simulation type ("MH" or "SLICE"); variable: the model parameter for which the simulation parameters are being overridden; Outcome.Grp (if applicable); Outcome (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value. The function *bhpm.sim.control.params* generates a template for *sim.params* which can be edited by the user.
`initial_values`	The initial values for starting the chains. If NULL (the default) is passed the function generates the initial values for the chains. initial_values is a list with the following format: `list(gamma, theta, mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi, mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0, alpha.pi, beta.pi)` The function *bhpm.gen.initial.values* can be used to generate a template for the list which can be updated by the user if required.

`level`	Allowed valus are 0, 1, 2. Respectively these indicate independent clusters, common means across the clusters and weak relationships between the clusters.
`hyper_params`	The hyperparameters for the model.
`global.pm.weight`	A global weighting for the proposal distribution used to sample theta.
`pm.weights`	Override global.pm.weight for specific outcomes.
`adapt_phase`	Unused parameter.
`memory_model`	Allowed values are "HIGH" and "LOW". "HIGH" means use as much memory as possible. "LOW" means use the minimum amount of memory.

Details

The model is fitted by a Gibbs sampler.

Value

The output from the simulation including all the sampled values is as follows:

```
list(id, theta_alg, sim_type, chains, nClusters, Clusters, nOutcome.Grp,
maxOutcome.Grps, maxOutcomes, nAE, AE, B, burnin,
iter, monitor, mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0,
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi, alpha.pi, beta.pi,
alpha.pi_acc, beta.pi_acc, gamma, theta, gamma_acc, theta_acc)
```

where

id - a string identifying the verions of the function.

theta_alg - an string identifying the algorithm used to smaple theta.

sim_type - an string identifying the samlping method used for non-standard distributions, either *"MH"* or *"SLICE"*.

chains - the number of chains for which the simulation was run.

nClusters - the number of clusters in the simulation.

Clusters - an array. The clusters.

nOutcome.Grp - the number of outcome groupings.

maxOutcome.Grps - the maximum number of outcome groupings in a cluster.

maxOutcomes - the maximum number of outcome in a outcome grouping.

nOutcome - an array. The number of outcomes in each outcome grouping.

Outcome - an array of dimension *nOutcome.Grp*, *maxOutcomes*. The outcomes.

Outcome.Grp - an array. The outcome groupings.

burnin - burnin used for the simulation.

iter - the total number of iterations in the simulation.

monitor - the variables being monitored. A dataframe.

mu.gamma.0 - array of generated samples.

mu.theta.0 - array of generated samples.

tau2.gamma.0 - array of generated samples.

tau2.theta.0 - array of generated samples.

mu.gamma - array of generated samples.

mu.theta - array of generated samples.

sigma2.gamma - array of generated samples.

sigma2.theta - array of generated samples.

pi - array of generated samples. *alpha.pi* - array of generated samples. *beta.pi* - array of generated samples.

alpha.pi_acc - the acceptance rate for the alpha.pi samples if a Metropolis-Hastings method is used.

beta.pi_acc - the acceptance rate for the beta.pi samples if a Metropolis-Hastings method is used.

gamma - array of generated samples.

theta - array of generated samples.

gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used.

theta_acc - the acceptance rate for the theta samples.

Note

The function performs the simulation and returns the raw output. No checks for convergence are performed.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
raw = bhpm.cluster.BB.hier3(bhpm.cluster.data1, level = 1, , burnin = 100, iter = 200)


data(bhpm.cluster.data1)
raw = bhpm.cluster.BB.hier3(bhpm.cluster.data1, level = 1)
```

---

bhpm.cluster.data1 *Cluster analysis data.*

Description

This dataset contains counts of the outcomes and exposure times for two treatment groups and six clusters.

Usage

```
data(bhpm.cluster.data1)
```

Format

A dataframe with columns *Cluster*, *Outcome.Grp*, *Outcoome*, *Trt.Grp*, *Count*, *Exposure*. The dataframe contains 1860 observations.

---

bhpm.cluster.data2 *Cluster analysis data.*

Description

This dataset contains counts of the coutcome and exposures for two treatments and ten clusters.

Usage

```
data(bhpm.cluster.data2)
```

Format

A dataframe with columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp*, *Count*, *Exposure*. The dataframe contains 1860 observations.

---

bhpm.convergence.diag *Convergence Diagnostics of the Simulation*

Description

The function applies either Gelman-Rubin or the Geweke diagnostic to the raw output of model simulation (e.g. bhpm.cluster.BB.hier3). It returns the convergence diagnostics and, if applicable, the acceptance rates for the sampling distributions.

Usage

```
bhpm.convergence.diag(raw, debug_diagnostic = FALSE)
```

Arguments

`raw`	The output from a model simulation.
`debug_diagnostic`	Unused parameter.

Details

This function applies one of two convergence diagnostics to the raw output of a model simulation in order to allow convergence to be assessed. The two diagnotics are:

i) Gelman-Rubin diagnostic - used when there is more than one chain. A value close to 1 is consistent with an MCMC simulation which has converged. The 'coda' diagnostic returns a point estimate and upper confidence limits.

ii) Geweke diagnostic - used when there is a single chain. A Z-score which is consistent with a standard normal distribution is expected from an MCMC simulation which has converged.

The raw sample data is converted to 'coda' format (mcmc objects) and the 'coda' methods gelman.diag and geweke.diag are used to perform the checks.

Value

Returns a list of the diagnostics for each sampled variable. Each individual element of the list is a data.frame containing at least the columns *type*, which is the type of diagnostic ('Gelman-Rubin' or 'Geweke'), *stat*, which is the value of the dignostic, and *upper_ci* which is the upper confidence interval for the Gelman-Rubin diagnostic. For the Geweke diagnostic *upper_ci* contains the value NA. Depending on the simulation performed the return from *bhpm.convergence.diag* will contain different variables. Columns which may be used to indentify the individual samples are *Trt.Grp*, *Outcome.Grp*, *Outcome*, and *Cluster*.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
data <- subset(bhpm.cluster.data1, Cluster == '0.0-180.0' | Cluster == '180.0-360.0')
raw = bhpm.npm(data, burnin = 100, iter = 200)
conv = bhpm.convergence.diag(raw)


data(bhpm.cluster.data1)
raw = bhpm.npm(bhpm.cluster.data1)
conv = bhpm.convergence.diag(raw)
```

---

bhpm.gen.initial.values

Generate a template simulation initial values.

Description

This function generates a template for the initial values to be used to start the simulation. They can be updated by the caller and passed to the simulation function.

Usage

```
bhpm.gen.initial.values(cluster.data, nchains = 3,
model = "1a", hier = 3, level = 1)
```

Arguments

cluster.data	A file or data frame containing the data for analysis.
nchains	The number of chains in the simulation.
model	The model type: "BB" for point-mass models, "1a" for non-point-mass models.
hier	Allowed values are 2, 3 indicating two or three level hierarchies.
level	The dependency level in the model: 0, 1, 2.

Value

A dataframe containing the template of intial values.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
initial.values <- bhpm.gen.initial.values(bhpm.cluster.data1)
print(initial.values$mu.gamma.0)

data(bhpm.cluster.data1)
initial.values <- bhpm.gen.initial.values(bhpm.cluster.data1)
print(initial.values$mu.gamma.0)
```

---

bhpm.global.sim.param.defaults

Generate default global simulation parameters for a model.

Description

Generate default global simulation parameters for a model.

Usage

```
bhpm.global.sim.param.defaults(model = "BB", hier = 3)
```

Arguments

model	Allowed values are "BB" and "1a" for point-mass and non-point-mass models respectively.
hier	Allowed values are 2 and 3 two-level and three-level hierarchies repsectively.

Value

A dataframe containing the global simulation parameters.

Author(s)

R. Carragher

Examples

```
g.s.p <- bhpm.global.sim.param.defaults("BB", 3)
print(g.s.p)

g.s.p <- bhpm.global.sim.param.defaults("BB", 3)
print(g.s.p)
```

---

bhpm.hyper.param.defaults

Generate default hyperparameter values for a model.

Description

Generate default hyperparameter values for a model.

Usage

```
bhpm.hyper.param.defaults(model = "BB", hier = 3)
```

Arguments

model	Allowed values are "BB" and "1a" for point-mass and non-point-mass models respectively.
hier	Allowed values are 2 and 3 two-level and three-level hierarchies repsectively.

Value

A list containing the default parameter values.

Author(s)

R. Carragher

Examples

```
h.p <- bhpm.hyper.param.defaults("BB", 3)
print(h.p$mu.gamma.0.0)

h.p <- bhpm.hyper.param.defaults("BB", 3)
print(h.p$mu.gamma.0.0)
```

bhpm.monitor.defaults *Generate default variable monitor list for a model.*

Description

Generate default variable monitor list for a model.

Usage

```
bhpm.monitor.defaults(model = "BB", hier = 3)
```

Arguments

model	Allowed values are "BB" and "1a" for point-mass and non-point-mass models respectively.
hier	Allowed values are 2 and 3 two-level and three-level hierarchies repsectively.

Value

A dataframe containing the default monitored variables.

Author(s)

R. Carragher

Examples

```
mon <- bhpm.monitor.defaults("BB", 3)
print(mon)

mon <- bhpm.monitor.defaults("BB", 3)
print(mon)
```

bhpm.monitor.samples *Generate a template for choosing which samples to monitor.*

Description

This function generate a template for choosing which samples to monitor based on the model and hierarchy. As some of the MCMC model simulations require large amounts of memory choosing not to monitor samples reduced the overall memory footprint.

Usage

```
bhpm.monitor.samples(model = "1a", hier = 3)
```

Arguments

model	Allowed values are "1a" and "BB". "BB" models include a point-mass.
hier	Allowed values are 2 and 3. Generate a template for a 2 or 3 level hierarchy.

Value

A dataframe containing two columns:

variable: the name of a class of variables e.g. "theta" *monitor*: 0 - don't monitor, 1 - monitor.

Author(s)

R. Carragher

Examples

```
mon <- bhpm.monitor.samples("1a", hier = 3)
print(mon)

mon <- bhpm.monitor.samples("1a", hier = 3)
print(mon)
```

---

bhpm.multi.treatments *Cluster analysis data.*

Description

This data set contains counts of outcomes and exposure times for four treatments over all six clusters of an observational study.

Usage

```
data(bhpm.multi.treatments)
```

Format

A dataframe with columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp*, *Count*, *Exposure*. The dataframe contains 3720 observations.

bhpm.multi.treatments.random.order *Cluster analysis data.*

Description

This data set contains count of outcome for four treatments over six clusters of an observational study. The data are in random order.

Usage

```
data(bhpm.multi.treatments.random.order)
```

Format

A dataframe with columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp*, *Count*, *Exposure*. The dataframe contains 3720 observations.

bhpm.npm *Fit a Bayesian Hierarchical Model for Grouped Data with Clusters and without Point-Mass.*

Description

Implementation of a Bayesian Hierarchical for Grouped Data with Clusters and without Point-Mass.

Usage

```
bhpm.npm(cluster.data, hier = 3, sim_type = "SLICE", burnin = 10000,
iter = 40000, nchains = 3,
global.sim.params = NULL,
sim.params = NULL,
monitor = NULL,
initial_values = NULL,
level = 1,
hyper_params = NULL,
memory_model = "HIGH")
```

Arguments

cluster.data	A file or data frame containing the cluster data. It must contain the columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp* (1 - control, 2,... comparator treatments), *Count* (total number of events), *Exposure* (total exposure of time of all patients for the Trt.Grp in the Cluster).
hier	Fit a 2 or 3 level hierarcical model.

`sim_type` The type of MCMC method to use for simulating from non-standard distributions. Allowed values are *"MH"* and *"SLICE"* for Metropis_Hastings and Slice sampling respectively.

`burnin` The burnin period for the monte-carlo simulation. These are discarded from the returned samples.

`iter` The total number of iterations for which the monte-carlo simulation is run. This includes the burnin period. The total number of samples returned is *iter - burnin*

`nchains` The number of independent chains to run.

`global.sim.params`
A data frame containing the parameters for the simuation type *sim_type*. For *"MH"* the parameter is the variance of the normal distribution used to simulate the next candidate value centred on the current value. For *"SLICE"* the parameters are the estimated width of the slice and a value limiting the search for the next sample. Passing NULL uses the model defaults.

`sim.params` A dataframe containing simulation parameters which override the global simulation parameters (*global.sim.params*) for particular model parameters. *sim.params* must contain the following columns: type: the simulation type ("MH" or "SLICE"); variable: the model parameter for which the simulation parameters are being overridden; Outcome.Grp (if applicable); Outcome (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

The function *bhpm.sim.control.params* generates a template for *sim.params* which can be edited by the user.

`monitor` A dataframe indicating which sets of variables to monitor. Passing NULL uses the model defaults.

`initial_values` The initial values for starting the chains. If NULL (the default) is passed the function generates the initial values for the chains. initial_values is a list with the following format:

```
list(gamma, theta, mu.gamma, mu.theta, sigma2.gamma,
sigma2.theta, mu.gamma.0, mu.theta.0, tau2.gamma.0,
tau2.theta.0)
```

where each element of the list is either a dataframe or array. The function *bhpm.gen.initial.values* can be used to generate a template for the list which can be updated by the user if required.

`level` The level of dependancy between the clusters. 0 - independent clusters, 1 - common cluster means, 2 - weak dependancy between clusters.

`hyper_params` The hyperparameters for the model. Passing NULL uses the model defaults.

`memory_model` Allowed values are "HIGH" and "LOW". "HIGH" means use as much memory as possible. "LOW" means use the minimum amount of memory.

Details

The model is fitted by a Gibbs sampler. The posterior distributions for *gamma* and *theta* are sampled with either a Metropolis-Hastings step or a slice sampler.

Value

The output from the simulation including all the sampled values is as follows:

```
list(id, sim_type, chains, nClusters, Clusters, nOutcome.Grp, maxOutcome.Grps,
maxOutcomes, nOutcome, Outcome, Outcome.Grp, burnin, iter, monitor,
mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0,
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, gamma,
theta, gamma_acc, theta_acc)
```

where

id - a string identifying the verion of the function

sim_type - an string identifying the samlping method used for non-standard distributions, either *"MH"* or *"SLICE"*

chains - the number of chains for which the simulation was run.

nClusters - the number of clusters in the simulation

Clusters - an array. The clusters.

nOutcome.Grp - the number of outcome groupings.

maxOutcome.Grps - the maximum number of outcome groupings in a cluster.

maxOutcomes - the maximum number of outcomes in an outcome grouping.

nOutcome - an array. The number of outcomes in each outcome grouping.

Outcome - an array of dimension *nOutcome.Grp*, *maxOutcomes*. The outcomes.

Outcome.Grp - an array. The outcome groupings.

burnin - burnin used for the simulation.

iter - the total number of iterations in the simulation.

monitor - the variables being monitored. A dataframe.

mu.gamma.0 - array of generated samples.

mu.theta.0 - array of generated samples.

tau2.gamma.0 - array of generated samples.

tau2.theta.0 - array of generated samples.

mu.gamma - array of generated samples.

mu.theta - array of generated samples.

sigma2.gamma - array of generated samples.

sigma2.theta - array of generated samples.

gamma - array of generated samples.

theta - array of generated samples.

gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used.

theta_acc - the acceptance rate for the theta samples if a Metropolis-Hastings method is used.

Note

The function performs the simulation and returns the raw output. No checks for convergence are performed.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
raw = bhpm.npm(cluster.data = bhpm.cluster.data1, burnin = 100, iter = 200)

data(bhpm.cluster.data1)
raw = bhpm.npm(cluster.data = bhpm.cluster.data1, level = 1)
```

---

bhpm.pm *A Bayesian Hierarchical Model for grouped data and clusters with Point-Mass.*

Description

Implementation of a bayesian Hierarchical for grouped data and clusters with Point-Mass.

Usage

```
bhpm.pm(cluster.data, hier = 3, sim_type = "SLICE", burnin = 20000,
iter = 60000, nchains = 5, theta_algorithm = "MH",
global.sim.params = NULL,
sim.params = NULL,
monitor = NULL,
initial_values = NULL, level = 1, hyper_params = NULL,
global.pm.weight = 0.5,
pm.weights = NULL,
adapt_phase=1, memory_model = "HIGH")
```

Arguments

cluster.data	A file or data frame containing the cluster data. It must contain the columns *Cluster*, *Outcome.Grp*, *Outcome*, *Trt.Grp* (1 - control, 2,... comparator treatments), *Count* (total number of events), *Exposure* (total exposure of time of all patients fot the Trt.Grp in the Cluster).
hier	Fit a 2 or 3 level model.
burnin	The burnin period for the monte-carlo simulation. These are discarded from the returned samples.

`iter` The total number of iterations for which the monte-carlo simulation is run. This includes the burnin period. The total number of samples returned is *iter - burnin*

`nchains` The number of independent chains to run.

`theta_algorithm`
MCMC algorithm used to sample the theta variables. "MH" is the only currently supported stable algorithm.

`sim_type` The type of MCMC method to use for simulating from non-standard distributions apart from theta. Allowed values are *"MH"* and *"SLICE"* for Metropis_Hastings and Slice sampling respectively.

`monitor` A dataframe indicating which sets of variables to monitor. Passing NULL uses the model defaults.

`global.sim.params`
A data frame containing the parameters for the simuation type *sim_type*. For *"MH"* the parameter is the variance of the normal distribution used to simulate the next candidate value centred on the current value. For *"SLICE"* the parameters are the estimated width of the slice and a value limiting the search for the next sample. Passing NULL uses the model defaults.

`sim.params` A dataframe containing simulation parameters which override the global simulation parameters (*global.sim.params*) for particular model parameters. *sim.params* must contain the following columns: type: the simulation type ("MH" or "SLICE"); variable: the model parameter for which the simulation parameters are being overridden; Outcome.Grp (if applicable); Outcome (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

The function *bhpm.sim.control.params* generates a template for *sim.params* which can be edited by the user.

`initial_values` The initial values for starting the chains. If NULL (the default) is passed the function generates the initial values for the chains. initial_values is a list with the following format:

```
list(gamma, theta, mu.gamma, mu.theta, sigma2.gamma,
sigma2.theta, pi, mu.gamma.0, mu.theta.0,
tau2.gamma.0, tau2.theta.0, alpha.pi, beta.pi)
```

The function *bhpm.gen.initial.values* can be used to generate a template for the list which can be updated by the user if required.

`level` Allowed valus are 0, 1, 2. Respectively these indicate independent clusters, common means across the clusters and weak relationships between the clusters.

`hyper_params` The hyperparameters for the model. Passing NULL uses the model defaults.

`global.pm.weight`
A global weighting for the proposal distribution used to sample theta.

`pm.weights` Override global.pm.weight for specific outcomes.

`adapt_phase` Unused parameter.

`memory_model` Allowed values are "HIGH" and "LOW". "HIGH" means use as much memory as possible. "LOW" means use the minimum amount of memory.

Details

The model is fitted by a Gibbs sampler.

Value

The output from the simulation including all the sampled values is as follows:

```
list(id, theta_alg, sim_type, chains, nClusters, Clusters, nOutcome.Grp,
maxOutcome.Grps, maxOutcomes, nAE, AE, B, burnin,
iter, monitor, mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0,
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi, alpha.pi, beta.pi,
alpha.pi_acc, beta.pi_acc, gamma, theta, gamma_acc, theta_acc)
```

where

id - a string identifying the verions of the function.

theta_alg - an string identifying the algorithm used to smaple theta.

sim_type - an string identifying the samlping method used for non-standard distributions, either *"MH"* or *"SLICE"*.

chains - the number of chains for which the simulation was run.

nClusters - the number of clusters in the simulation.

Clusters - an array. The clusters.

nOutcome.Grp - the number of outcome groupings.

maxOutcome.Grps - the maximum number of outcome groupings in a cluster.

maxOutcomes - the maximum number of outcome in a outcome grouping.

nOutcome - an array. The number of outcomes in each outcome grouping.

Outcome - an array of dimension *nOutcome.Grp*, *maxOutcomes*. The outcomes.

Outcome.Grp - an array. The outcome groupings.

burnin - burnin used for the simulation.

iter - the total number of iterations in the simulation.

monitor - the variables being monitored. A dataframe.

mu.gamma.0 - array of generated samples.

mu.theta.0 - array of generated samples.

tau2.gamma.0 - array of generated samples.

tau2.theta.0 - array of generated samples.

mu.gamma - array of generated samples.

mu.theta - array of generated samples.

sigma2.gamma - array of generated samples.

sigma2.theta - array of generated samples.

pi - array of generated samples. *alpha.pi* - array of generated samples. *beta.pi* - array of generated samples.

alpha.pi_acc - the acceptance rate for the alpha.pi samples if a Metropolis-Hastings method is used.

beta.pi_acc - the acceptance rate for the beta.pi samples if a Metropolis-Hastings method is used.

gamma - array of generated samples.

theta - array of generated samples.

gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used.

theta_acc - the acceptance rate for the theta samples.

Note

The function performs the simulation and returns the raw output. No checks for convergence are performed.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
raw = bhpm.pm(cluster.data = bhpm.cluster.data1, burnin = 100, iter = 200)


data(bhpm.cluster.data1)
raw = bhpm.pm(cluster.data = bhpm.cluster.data1)
```

---

bhpm.pointmass.weights

Generate a template for the point-mass weightings.

Description

This function generate a template for weights for the proposal distribution used to sample *theta* variables in models which use a point-mass.

Usage

```
bhpm.pointmass.weights(cluster.data)
```

Arguments

`cluster.data`	A file or data frame containing the cluster data for analysis.

Value

A dataframe containing the weightings template for each outcome grouping, outcome and, if required, cluster.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
pmw <- bhpm.pointmass.weights(bhpm.cluster.data1)
head(pmw, 2)
```

---

bhpm.print.convergence.summary

Print a Summary of the Convergence Diagnostics of the Simulation

---

Description

The function prints the maximum and minimum values of either Gelman-Rubin diagnostic or the Geweke diagnostic for each group of samples, e.g. theta, gamma, mu.gamma etc.

Usage

```
bhpm.print.convergence.summary(conv)
```

Arguments

conv	The output from a call to *bhpm.check.convergence*.

Value

Nothing

Note

The Geweke statistic is a Z-score calculated from a single chain. Due to the large number of variables sampled it is possible that a certain number will be deemed significant (at the 5% level) even though the simulation may have converged.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
data <- subset(bhpm.cluster.data1, Cluster == '0.0-180.0' | Cluster == '180.0-360.0')
raw = bhpm.npm(data, burnin = 100, iter = 200)
conv = bhpm.convergence.diag(raw)
bhpm.print.convergence.summary(conv)

data(bhpm.cluster.data1)
raw = bhpm.npm(bhpm.cluster.data1)
conv = bhpm.convergence.diag(raw)
bhpm.print.convergence.summary(conv)
```

---

bhpm.print.summary.stats

Print the Summary Statistics of Posterior Distributions

Description

The function prints the variable names, the mean, the HPI interval, the standard distribtion and the MCMC standard error for the simulated sample.

Usage

```
bhpm.print.summary.stats(summ)
```

Arguments

summ	The output from a call to bhpm.summary.stats

Value

Nothing

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
data <- subset(bhpm.cluster.data1, Cluster == '0.0-180.0')
raw = bhpm.npm(data, burnin = 100, iter = 200)
summ = bhpm.summary.stats(raw)
bhpm.print.summary.stats(summ)


data(bhpm.cluster.data1)
raw = bhpm.npm(bhpm.cluster.data1)
```

```
summ = bhpm.summary.stats(raw)
bhpm.print.summary.stats(summ)
```

---

bhpm.ptheta *Reports the posterior probability that theta (the increase in the log-odds) is greater than zero, zero, and less than zero for each outcome*

Description

This function reports the posterior probability that theta is positive negative or zero, i.e. that there is an increase, decrease, or no difference in the log odds of an outcome being associated with treatment.

Usage

```
bhpm.ptheta(raw)
```

Arguments

raw	The output from a call to one of bhpm.cluster.BB.hier3, bhpm.cluster.1a.hier3, bhpm.cluster.BB.hier2, bhpm.cluster.1a.hier2.

Value

A data frame containing the columns: *Trt.Grp*, *Cluster*, *Outcome.Grp*, *Outcome*, *ptheta*, *ptheta.pos*, *ptheta.zero*, *ptheta.neg*.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
data <- subset(bhpm.cluster.data1, Cluster == '0.0-180.0' | Cluster == '180.0-360.0')
raw = bhpm.npm(data, burnin = 10, iter = 100)
p = bhpm.ptheta(raw)
head(p, 2)
```

```
data(bhpm.cluster.data1)
raw = bhpm.npm(bhpm.cluster.data1)
p = bhpm.ptheta(raw)
head(p, 2)
```

bhpm.sim.control.params *Generate a template for the individual model parameter simulation control parameters.*

Description

This function generates a template for overriding the global simulation parameters used by the model simulation functions (e.g. bhpm.cluster.BB.hier3).

Usage

```
bhpm.sim.control.params(cluster.data, model = "1a")
```

Arguments

cluster.data	A file or data frame containing the cluster data.
model	Allowed values are "BB" and "1a" for point-mass and non-point-mass models respectively.

Value

A dataframe containing the simulation parameters template.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
s.c.p <- bhpm.sim.control.params(bhpm.cluster.data1)
head(s.c.p)

data(bhpm.cluster.data1)
s.c.p <- bhpm.sim.control.params(bhpm.cluster.data1)
head(s.c.p)
```

bhpm.summary.stats *Summary Statistics for the Posterior Distributions in the model.*

Description

Returns the Summary Statistics for the Posterior Distributions in the model.

Usage

```
bhpm.summary.stats(raw, prob)
```

Arguments

raw	The output from a model simulation (e.g. bhpm.cluster.BB.hier3).
prob	The probability level for HPI. Default is 0.95.

Details

The function reports the mean, upper and lower bounds of the HPI (highest probabily interval), the standard deviation and MCMC standard error.

Value

Returns a list of the summary statistics for each sampled variable. Each element of the list is a data.frame containing at least the columns *mean*, *median*, *hpi_lower*, *hpi_upper*, *SD* and *SE*. Columns which may be used to indentify the individual variables are *Outcome.Grp*, *Outcome*, and *Cluster*.

Note

The MCMC error is found using the 'coda' summary function.

Author(s)

R. Carragher

Examples

```
data(bhpm.cluster.data1)
data <- subset(bhpm.cluster.data1, Cluster == '0.0-180.0')
raw = bhpm.npm(data, burnin = 100, iter = 200)
summ = bhpm.summary.stats(raw)


data(bhpm.cluster.data1)
raw = bhpm.npm(bhpm.cluster.data1)
summ = bhpm.summary.stats(raw)
```

Index