Package ‘c212’

October 8, 2018

Type Package
Title Methods for Detecting Safety Signals in Clinical Trials Using Body-Systems (System Organ Classes)
Version 0.94
Date 2018-10-05
Author Raymond Carragher <raymond.carragher@strath.ac.uk>
Maintainer Raymond Carragher <raymond.carragher@strath.ac.uk>
Depends coda
Encoding UTF-8
Description Methods for detecting safety signals in clinical trials using groupings of adverse events by body-system or system organ class. The package title c212 is in reference to the original Engineering and Physical Sciences Research Council (UK) funded project which was named CASE 2/12.
License GPL (>= 2)
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-10-08 09:50:11 UTC

R topics documented:

c212-package .......................................................... 2
c212.1a .............................................................. 4
c212.BB .............................................................. 7
c212.BH .............................................................. 12
c212.BH.adjust.pvals ................................................ 14
c212.bin.test ......................................................... 16
c212.BONF .......................................................... 17
c212.convergence.diag ............................................. 18
c212.DFDR ........................................................... 20
c212.FDR.data ....................................................... 21
c212.fisher.test .................................................... 21
c212.GBH ........................................................... 23
Description

This package implements a number of methods for the detection of safety signals in Clinical Trials based on groupings of adverse events by body-system or system organ class. The methods include an implementation of the Three-Level Hierarchical model for Clinical Trial Adverse Event Incidence Data of Berry and Berry (2004) and an implementation of the same model without the Point Mass (Model 1a from Xia et al (2011)), extended Bayesian hierarchical methods based on system organ class or body-system groupings for interim analyses. The package also implements a number of methods for error control when testing multiple hypotheses, specifically control of the False Discovery Rate (FDR). The FDR control methods implemented are the Benjamini-Hochberg procedure, the Double False Discovery Rate, the Group Benjamini-Hochberg and subset Benjamini-Hochberg methods. Also included are the Bonferroni correction and the unadjusted testing procedure.
Details

The methods implemented use assumed groupings of adverse events by body-system or system organ class to detect differences in the occurrence of adverse events on trial arms. Methods based on Bayesian Hierarchical models and direct error controlling procedures are provided.

The basic (Bayesian) hierarchical models are described in Berry and Berry (2004), Xia et al (2011) (Model 1a) and Berry et al (2010). These methods are extended for interim analyses.

The direct error controlling methods are designed to control the number of Type-I errors at an acceptable level without compromising the power. If the Familywise Error Rate (FWER) is defined as the probability of making one or more Type-I errors when analysing multiple hypotheses (the “family”), then an alternative to controlling the FWER is to control the False Discovery Rate (FDR) - the expected proportion of false discoveries (Type-I errors) to the total number of discoveries. Essentially control of the FDR assumes that when many of the tested hypotheses are rejected it may be preferable to control the proportion of errors rather than the probability of making even one error. This is expected to lead to a gain in power. Further FDR controlling methods which use the information available in groupings of hypotheses have been developed (Double False Discovery Rate (Mehrotra and Adewale (2012)), Group Benjamini-Hochberg (Hu, Zhao and Zhou (2010))). For the methods contained in this package control of the False Discovery Rate has been established for independent test statistics and some forms of positive dependency (positive regression dependency), apart from the case of the Group Benjamini-Hochberg procedure where the control is asymptotic. Further details can be found in the references.

Author(s)

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

References


---

Implementation of the Berry and Berry Three-Level Hierarchical Model without Point-Mass.

**Description**

Implementation of Berry and Berry model without the point-mass (Model 1a Xia et al (2011))

**Usage**

```r
C212.1a(trial.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(0.35,1), control = c(0,6), stringsAsFactors = FALSE), 
  sim.params = NULL, 
  initial_values = NULL, 
  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))
```

**Arguments**

- **trial.data**: A file or data frame containing the trial data. It must contain the columns `B` (body-system), `AE` (adverse event), `Group` (1 - control, 2 treatment), `Count` (total number of events), `Total` (total number of participants in the trial arm).

- **sim_type**: The type of MCMC method to use for simulating from non-standard distributions. Allowed values are "MH" and "SLICE" for Metropolis_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 simulation 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 data frame 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; B: the body-system (if applicable); AE: the adverse event (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.
The function c212.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, 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 c212.gen.initial.values can be used to generate a template for the list which can be updated by the user if required. The formats of the list elements are as follows:
gamma, theta: dataframe with columns B, AE, chain, value
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta: dataframe with columns B, chain, value
mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0: array of size chain.

hyper_params
The hyperparameters for the model. The default hyperparameters are those given in Berry and Berry 2004.

Details
The model is fitted by a Gibbs sampler. The details of the complete conditional densities are given in Berry and Berry (2004). 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, nBodySys, maxAEs, nAE, AE, B, burnin, iter, 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 version of the function

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

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

*nBodySys* - the number of body-systems

*maxAEs* - the maximum number of AEs in a body-system

*nAE* - an array. The number of AEs in each body-system.

*AE* - an array of dimension *nBodySys*, *maxAEs*. The Adverse Events.

*B* - an array. The body-systems.

*burnin* - the burnin period for the simulation.

*iter* - the total number of iterations in the simulation.

*mu.gamma.0* - array of samples of dimension *chains*, *iter* - *burnin*

*mu.theta.0* - array of samples of dimension *chains*, *iter* - *burnin*

*tau2.gamma.0* - array of samples of dimension *chains*, *iter* - *burnin*

*tau2.theta.0* - array of samples of dimension *chains*, *iter* - *burnin*

*mu.gamma* - array of samples of dimension *chains*, *nBodySys* *iter* - *burnin*

*mu.theta* - array of samples of dimension *chains*, *nBodySys* *iter* - *burnin*

*sigma2.gamma* - array of samples of dimension *chains*, *nBodySys* *iter* - *burnin*

*sigma2.theta* - array of samples of dimension *chains*, *nBodySys* *iter* - *burnin*

*gamma* - array of samples of dimension *chains*, *nBodySys*, *maxAEs*, *iter* - *burnin*

*theta* - array of samples of dimension *chains*, *nBodySys*, *maxAEs*, *iter* - *burnin*

*gamma_acc* - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used. An array of dimension *chains*, *nBodySys*, *maxAEs*

*theta_acc* - the acceptance rate for the theta samples if a Metropolis-Hastings method is used. An array of dimension *chains*, *nBodySys*, *maxAEs*

**Note**

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

**Author(s)**

R. Carragher
References


Examples

```r
# Not run:
data(c212.trial.data)
raw = c212.1a(c212.trial.data)

raw$B
[1] "Bdy-sys_1" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5" "Bdy-sys_6"
[7] "Bdy-sys_7" "Bdy-sys_8"

mean(rm$theta[2, 3, 1,])
[1] 1.306362

# End(Not run)
```
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.params = data.frame(min_w = 0.25, chains = 3, burnin = 20000,
iter = 40000),
adapt.phase=0)

Arguments

trial.data   A file or data frame containing the trial data. It must contain columns B (body-system), AE (adverse event), Group (1 - control, 2 treatment), Count (total number of events), Total (total number of participants).
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 Metropolis_Hastings and Slice sampling respectively.
global.sim.params    A data frame containing the parameters for the simulation 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; B: the body-system (if applicable); AE: the adverse event (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.
The function c212.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 `c212.gen.initial.values` can be used to generate a template for the list which can be updated by the user if required. The formats of the list elements are as follows:

- *gamma, theta*: dataframe with columns `B, AE, chain, value`
- *mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi*: dataframe with columns `B, chain, value`
- *mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0, alpha.pi, beta.pi*: array of size `chain`.

**hyper_params**

The hyperparameters for the model. The default hyperparameters are those given in Berry and Berry 2004.

- *global.pm.weight*
  A global weighting for the proposal distribution used to sample theta.
- *pm.weights*
  Override global.pm.weight for specific adverse events.
- *adapt_params*
  Unused parameter.
- *adapt_phase*
  Unused parameter.

**Details**

The model is fitted by a Gibbs sampler. The details of the complete conditional densities are given in Berry and Berry (2004).

**Value**

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

```
list(id, theta_alg, sim_type, chains, nBodySys, maxAEs, nAE, AE, B, burnin, iter, 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, theta_zero_prop, theta_zero_acc)
```

where

- *id* - a string identifying the versions of the function
- *theta_alg* - an string identifying the algorithm used to sample theta
- *sim_type* - an string identifying the sampling method used for non-standard distributions, either "MH" or "SLICE"
- *chains* - the number of chains for which the simulation was run
- *nBodySys* - the number of body-systems
- *maxAEs* - the maximum number of AEs in a body-system
- *nAE* - an array. The number of AEs in each body-system.
- *AE* - an array of dimension `nBodySys, maxAEs`. The Adverse Events.
- *B* - an array. The body-systems.
- *mu.gamma.0* - array of samples of dimension `chains, iter - burnin`
- *mu.theta.0* - array of samples of dimension `chains, iter - burnin`
tau2.gamma.0 - array of samples of dimension chains, iter - burnin

tau2.theta.0 - array of samples of dimension chains, iter - burnin

mu.gamma - array of samples of dimension chains, nBodySys iter - burnin

mu.theta - array of samples of dimension chains, nBodySys iter - burnin

sigma2.gamma - array of samples of dimension chains, nBodySys iter - burnin

sigma2.theta - array of samples of dimension chains, nBodySys iter - burnin

pi - array of samples of dimension chains, nBodySys iter - burnin

alpha.pi - array of samples of dimension chains, iter - burnin

beta.pi - array of samples of dimension chains, iter - burnin

alpha.pi_acc - the acceptance rate for the alpha.pi samples if a Metropolis-Hastings method is used. An array of dimension chains, maxAEs

beta.pi_acc - the acceptance rate for the beta.pi samples if a Metropolis-Hastings method is used. An array of dimension chains, maxAEs

gamma - array of samples of dimension chains, nBodySys, maxAEs, iter - burnin

theta - array of samples of dimension chains, nBodySys, maxAEs, iter - burnin

gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used. An array of dimension chains, nBodySys, maxAEs

theta_acc - the acceptance rate for the theta samples. An array of dimension chains, nBodySys, maxAEs

theta_zero_prop - the number of zeros proposed in theta sampling. An array of dimension chains, nBodySys, maxAEs

theta_zero_acc - the acceptance rate for zeros for the theta samples. An array of dimension chains, nBodySys, maxAEs

Note

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

Author(s)

R. Carragher

References


Examples

```r
## Not run:

data(c212.trial.data)
raw = c212.BB(c212.trial.data)

raw$B
[1] "Bdy-sys_1" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5" "Bdy-sys_6"
[7] "Bdy-sys_7" "Bdy-sys_8"

mean(raw$theta[2, 1, 1])
[1] 0.1088401
```
Implementation of Benjamini-Hochberg procedure for False Discovery Rate control

Description

Implementation of Benjamini-Hochberg procedure for False Discovery Rate control. The hypotheses' data can be contained in a file or data frame.

Usage

c212.BH(trial.data, alpha = 0.05)

Arguments

trial.data File or data frame containing the p-values for the hypotheses being tested. The data must include a column called p which contains the p-values of the hypotheses.

alpha The level for FDR control. E.g. 0.05.

Value

The subset of hypotheses in file or trial.data deemed significant by the Benjamini-Hochberg procedure.

Note

No check is made for duplicate rows in the input file or data frame.

Author(s)

R. Carragher<raymond.carragher@strath.ac.uk>

References

Examples

```r
## Not run:
File: trial.dat

B j AE p
1 1 AE1 0.135005
2 1 AE2 0.010000
2 2 AE3 0.001000
2 3 AE4 0.005000
2 4 AE5 0.153501
3 1 AE6 0.020000
3 2 AE7 0.001300
3 3 AE8 0.002300
3 4 AE9 0.011000
3 5 AE10 0.023000
3 6 AE11 0.016000
3 7 AE12 0.010900
4 1 AE13 0.559111
4 2 AE14 0.751966
4 3 AE15 0.308339
4 4 AE16 0.837154
4 5 AE17 0.325882
c212.BH("trial.dat", alpha = 0.05)

B j AE p
1 2 2 AE3 0.0010
2 3 2 AE7 0.0013
3 3 3 AE8 0.0023
4 2 3 AE4 0.0050
5 2 1 AE2 0.0100
6 3 7 AE12 0.0109
7 3 4 AE9 0.0110
8 3 6 AE11 0.0160
9 3 1 AE6 0.0200
10 3 5 AE10 0.0230

## End(Not run)

trial.data <- data.frame(B = c(1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4),
                         j = c(1, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5),
                                "AE12", "AE13", "AE14", "AE15", "AE16", "AE17"),
                         p = c(0.135005, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023,
                              0.011, 0.023000, 0.016, 0.0109, 0.559111, 0.751966, 0.308339, 0.837154, 0.325882))
c212.BH(trial.data, 0.05)
```
Benjamini-Hochberg procedure adjusted p-values.

Usage

c212.BH.adjust.pvals(trial.data)

Arguments

trial.data File or data frame containing the p-values for the hypotheses being tested. The data must include a column called p which contains the p-values of the hypotheses.

Value

Returns the original data frame, ordered by p, with an additional column p_adj.

Note

The adjusted p values may be directly compared to a value alpha to determine whether to declare a hypothesis significant under the Benjamini-Hochberg procedure at level alpha.

Note

No check is made for duplicate rows in the input file or data frame.

Author(s)

R. Carragher<raymond.carragher@strath.ac.uk>
References

Examples

```r
trial.data <- data.frame(B = c(1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4),
                        j = c(1, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5),
                        p = c(0.135005, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023, 0.011, 0.023000, 0.016, 0.0109, 0.559111, 0.751986, 0.308339, 0.837154, 0.325882))

adj <- c212.BH.adjust.pvals(trial.data)

## Not run:
adj:
adj:
====
   B  j  AE      p  p_adj
 1  2  2  AE1  0.001000  0.01105000
 2  3  2  AE2  0.001300  0.01105000
 3  3  3  AE3  0.002300  0.01303333
 4  2  3  AE4  0.005000  0.02125000
 5  2  1  AE5  0.010000  0.02671429
 6  3  7  AE6  0.010900  0.02671429
 7  3  4  AE7  0.011000  0.02671429
 8  3  6  AE8  0.016000  0.03400000
 9  3  1  AE9  0.020000  0.03777778
10 3  5  AE10  0.023000  0.03910000
11 1  1  AE11  0.135005  0.20864409
12 2  4  AE12  0.153501  0.21745975
13 4  3  AE13  0.308339  0.39571386
14 4  5  AE14  0.325882  0.39571386
15 4  1  AE15  0.559111  0.63365913
16 4  2  AE16  0.751986  0.79898513
17 4  4  AE17  0.837154  0.83715400

## End(Not run)
adj[adj$p_adj <= 0.05, ]

## Not run:
B  j  AE      p  p_adj
 1  2  2  AE1  0.0010  0.01105000
 2  3  2  AE2  0.0013  0.01105000
 3  3  3  AE3  0.0023  0.01303333
 4  2  3  AE4  0.0050  0.02125000
 5  2  1  AE5  0.0100  0.02671429
 6  3  7  AE6  0.0109  0.02671429
```
c212.bin.test

Plot Raw Adverse Event Incidence Data

Description

Test the hypothesis that the proportions in two groups are the same. adverse event.

Usage

c212.bin.test(trial.data, alternative = "two.sided", correct = TRUE)

Arguments

trial.data A file or data frame containing the trial data. The data frame must contain the
columns B (body-system), AE (adverse event), Group (1 - control, 2 treatment),
Count (total number of events), Total (total number of participants).

alternative Alternative hypothesis may be "two.sided", "greater" or "less". The default is
"two.sided".

correct Apply a continuity correction.

Details

Test the hypothesis that the proportions in two groups are the same.

Value

Dataframe containing the results of the test. A copy of the input dataframe with an additional
column p containing the p-value from the test.

Note

Wrapper for the R function 'prop.test' in package 'stats'.

Author(s)

R. Carragher
Examples

```r
## Not run:
data(c212.trial.data)
pr = c212.bin.test(c212.trial.data)
head(pr)
   B  j   AE   p
1 Bdy-sys_1 Adv-Ev_1 3.176140e-01
2 Bdy-sys_2 Adv-Ev_2 1.011600e-02
3 Bdy-sys_2 Adv-Ev_3 2.687532e-02
4 Bdy-sys_2 Adv-Ev_4 6.714994e-01
5 Bdy-sys_2 Adv-Ev_5 7.590387e-01
6 Bdy-sys_3 Adv-Ev_6 1.000218e-06
```

## End(Not run)

---

### Description

The Bonferroni correction controls the Familywise Error Rate.

### Usage

```r
c212.BONF(trial.data, alpha = 0.05)
```

### Arguments

- `trial.data`: File or data frame containing the p-values for the hypotheses being tested. The data must include a column called `p` which contains the p-values of the hypotheses.
- `alpha`: The value for error control, e.g. 0.05.

### Value

The subset of hypotheses in `file` or `trial.data` deemed significant.

### Note

No check is made for duplicate rows in the input file or data frame.

### Author(s)

R. Carragher
References


Examples

```r
trial.data <- data.frame(B = c(1, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4),
                        j = c(1, 1, 2, 3, 4, 1, 2, 3, 4, 5),
                               "AE12", "AE13", "AE14", "AE15", "AE16", "AE17"),
                        p = c(0.135005, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023, 0.011,
                             0.023000, 0.016, 0.0109, 0.559111, 0.751906, 0.308339, 0.837154, 0.325882))

c212.BONF(trial.data, 0.05)
```

```r
## Not run:
B  j  AE  p
1 2 2 AE3 0.0010
2 3 2 AE7 0.0013
3 3 3 AE8 0.0023

## End(Not run)
```

---

**c212.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. c212.BB). It returns the convergence diagnostics and, if applicable, the acceptance rates for the sampling distributions.

**Usage**

```r
c212.convergence.diag(raw, debug_diagnostic = FALSE)
```

**Arguments**

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

parameter is time consuming. 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 diagnostics 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 diagnostic, 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 c212.convergence.diag will contain different variables. The return for a simulation from c212.1a is as follows:

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

Additional columns which may be used to indentify the individual samples are B, the body-system, and AE, the Adverse Event and interval.

Author(s)

R. Carragher

Examples

```r
## Not run:
data(c212.trial.data)
raw = c212.BB(c212.trial.data)
conv = c212.convergence.diag(raw)
## End(Not run)
```
**Implementation of the Double False Discovery Rate for controlling the False Discovery Rate.**

**Description**

The Double False Discovery Rate is designed to take advantage of possible groupings which may exist within sets of hypotheses. It applies the BH-procedure twice. Once at the group level, to identify sets of hypotheses which may contain significant hypotheses. It then groups these hypotheses together to form a single family and applies the BH-procedure again to declare hypotheses significant.

**Usage**

```r
c212.DFDR(trial.data, alpha = 0.05)
```

**Arguments**

- `trial.data`  
  File or data frame containing the p-values for the hypotheses being tested. The data must contain the following columns: 
  - `B`: the index or name of the groupings; 
  - `p`: the p-values of the hypotheses.

- `alpha`  
  The level for FDR control. E.g. 0.05.

**Value**

The subset of hypotheses in `file` or `trial.data` deemed significant by the Double False Discovery Rate process.

**Author(s)**

R. Carragher

**References**


**Examples**

```r
trial.data <- data.frame(B = c(1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4), 
                        p = c(0.135005, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023, 
                             0.011, 0.023000, 0.016, 0.0109, 0.559111, 0.751986, 0.308339, 0.837154, 0.325882))

c212.DFDR(trial.data, 0.05)
```
## Not run:

<table>
<thead>
<tr>
<th>B</th>
<th>j</th>
<th>AE</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>AE3</td>
<td>0.0010</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>AE7</td>
<td>0.0013</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>AE8</td>
<td>0.0023</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>AE4</td>
<td>0.0050</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>AE2</td>
<td>0.0100</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>AE12</td>
<td>0.0109</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>AE9</td>
<td>0.0110</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>AE11</td>
<td>0.0160</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>AE6</td>
<td>0.0200</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>AE10</td>
<td>0.0230</td>
</tr>
</tbody>
</table>

## End(Not run)

---

**c212.FDR.data**

*Fisher Test p-values for End of Trial Data Clinical Data*

### Description

This data set the p-values for a comparison between adverse events on each arm of a clinical trial.

### Usage

```r
c212.FDR.data
```

### Format

A dataframe with columns `B` - body-system, `AE` - adverse event, `p` - p-value for two-sided Fisher exact test. The dataframe contains 45 observations.

---

**c212.fisher.test**

*Fisher Exact Test*

### Description

Perform a Fisher exact test on clinical trial data.

### Usage

```r
c212.fisher.test(trial.data, alternative = "two.sided")
```
Arguments

- **trial.data**: A file or data frame containing the trial data. The data frame must contain the columns `B` (body-system), `AE` (adverse event), `Group` (1 - control, 2 treatment), `Count` (total number of events), `Total` (total number of participants).
- **alternative**: Alternative hypothesis may be "two.sided", "greater" or "less". The default is "two.sided".

Details

Perform a Fisher exact test on clinical trial data.

Value

Dataframe containing the results of the test. A copy of the input dataframe with an additional column `p` containing the p-value from the test.

Note

Wrapper for the R function 'fisher.test' in package 'stats'.

Author(s)

R. Carragher

Examples

```r
## Not run:
data(c212.trial.data)
f = c212.fisher.test(c212.trial.data)
head(f)
# B j AE p
# 1 Bdy-sys_1 1 Adv-Ev_1 2.892876e-01
# 2 Bdy-sys_2 1 Adv-Ev_2 5.333164e-03
# 3 Bdy-sys_2 2 Adv-Ev_3 1.601311e-02
# 4 Bdy-sys_2 3 Adv-Ev_4 6.502108e-01
# 5 Bdy-sys_2 4 Adv-Ev_5 7.437946e-01
# 6 Bdy-sys_3 1 Adv-Ev_6 3.746249e-08

## End(Not run)
```
Description

The Group Benjamini-Hochberg procedure for control of the False Discovery Rate is designed to take advantage of possible groupings which may exist within sets of hypotheses. The procedure estimates the number of true null hypotheses in each grouping and uses this to weight the p-values which are then compared to a weighted level for control. The procedure asymptotically controls the False Discovery Rate at the required level.

Usage

c212.GBH(trial.data, pi0 = "TST", alpha)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trial.data</td>
<td>File or data frame containing the p-values for the hypotheses being tested. The data must contain the following columns: B: the index or name of the groupings; p: the p-values of the hypotheses.</td>
</tr>
<tr>
<td>pi0</td>
<td>The estimator to use for the estimation of the number of true null hypotheses in each group. Valid values are &quot;TST&quot; and &quot;LSL&quot;.</td>
</tr>
<tr>
<td>alpha</td>
<td>The level for FDR control. E.g. 0.05.</td>
</tr>
</tbody>
</table>

Value

The subset of hypotheses in file or trial.data deemed significant by the Group Benjamini-Hochberg process.

Note

The estimator "TST" is the two-stage estimator of Benjamini, Krieger, and Yekutieli. The estimator "LSL" is the least-slope estimator of Benjamini and Hochberg.

Author(s)

R. Carragher

References


Examples

```r
trial.data <- data.frame(B = c(1, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4),
                      j = c(1, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5),
                              "AE12", "AE13", "AE14", "AE15", "AE16", "AE17"),
                      p = c(0.135005, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023,
                            0.011, 0.023000, 0.016, 0.0109, 0.559111, 0.751986, 0.308339, 0.837154, 0.325882))
```

c212.GBH(trial.data, pi = "TST", 0.05)

```r
## Not run:
B j AE p p_weighted
1 3 1 AE6 0.020000 0.000000000
2 3 2 AE7 0.001300 0.000000000
3 3 3 AE8 0.002300 0.000000000
4 3 4 AE9 0.011000 0.000000000
5 3 5 AE10 0.023000 0.000000000
6 3 6 AE11 0.016000 0.000000000
7 3 7 AE12 0.010900 0.000000000
8 2 2 AE3 0.001000 0.000333333
9 2 3 AE4 0.005000 0.001666667
10 2 1 AE2 0.010000 0.003333333
11 2 4 AE5 0.153501 0.051167000
## End(Not run)
```

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

```r
c212.gen.initial.values(trial.data, nchains = 3,
                      model = "1a", hier = 3, level = 0)
```

Arguments

- **trial.data**: A file or data frame containing the trial data, either for end of trial or interim
  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 initial values.

Author(s)

R. Carragher

Examples

```r
## Not run:
data(c212.trial.data)
c212.gen.initial.values(c212.trial.data)

## End(Not run)
```

Description


Usage

```r
c212.interim.1a.hier2(trial.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

**trial.data**
A file or data frame containing the trial data. It must contain the columns `I_index` (interval index), B (body-system), AE (adverse event), Group (1 - control, 2 treatment), Count (total number of events), Total (total number of participants in the trial arm).

**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 simulation 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; B: the body-system (if applicable); AE: the adverse event (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

The function `c212.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:

```r
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 `c212.gen.initial.values` can be used to generate a template for the list which can be updated by the user if required. The formats of the list elements are as follows:

- `gamma, theta`: dataframe with columns B, AE, chain, value
- `mu.gamma, mu.theta, sigma2.gamma, sigma2.theta`: dataframe with columns B, chain, value

**level**
The level of longitudinal dependancy between the intervals. 0 - independent intervals, 1 - common interva body-system means. dependancy.
The hyperparameters for the model. The default hyperparameters are those given in Berry and Berry 2004.

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:

\[
\text{list(id, sim\_type, chains, nIntervals, Intervals, nBodySys, maxBs, maxAEs, nAE, AE, B, burnin, iter, monitor,}
\mu.\gamma, \mu.\theta, \sigma_2.\gamma, \sigma_2.\theta, \gamma, \theta, gamma\_acc, theta\_acc)
\]

where

- \( id \) - a string identifying the version of the function
- \( sim\_type \) - an string identifying the sampling method used for non-standard distributions, either "MH" or "SLICE"
- \( chains \) - the number of chains for which the simulation was run.
- \( nIntervals \) - the number of intervals in the simulation
- \( Intervals \) - an array. The intervals.
- \( nBodySys \) - the number of body-systems
- \( maxBs \) - the maximum number of body-systems in an interval
- \( maxAEs \) - the maximum number of AEs in a body-system
- \( nAE \) - an array. The number of AEs in each body-system.
- \( AE \) - an array of dimension \( nBodySys, maxAEs \). The Adverse Events.
- \( B \) - an array. The body-systems.
- \( burnin \) - 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 samples of dimension \( chains, nBodySys \) iter - burnin
- \( \mu.\theta \) - array of samples of dimension \( chains, nBodySys \) iter - burnin
- \( \sigma_2.\gamma \) - array of samples of dimension \( chains, nBodySys \) iter - burnin
- \( \sigma_2.\theta \) - array of samples of dimension \( chains, nBodySys \) iter - burnin
- \( \gamma \) - array of samples of dimension \( chains, nBodySys, maxAEs, iter - burnin \)
- \( \theta \) - array of samples of dimension \( chains, nBodySys, maxAEs, iter - burnin \)
**gamma_acc** - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used. An array of dimension `chains, nBodySys, maxAEs`

**theta_acc** - the acceptance rate for the theta samples if a Metropolis-Hastings method is used. An array of dimension `chains, nBodySys, maxAEs`

**Note**

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

**Author(s)**

R. Carragher

**Examples**

```r
## Not run:
data(c212trial.interval.data1)
raw = c212.interim.1a.hier2(c212trial.interval.data1, level = 1)
raw$B

[1,] "Bdy-sys_1"  "Bdy-sys_10"  "Bdy-sys_11"  "Bdy-sys_12"  "Bdy-sys_13"
[2,] "Bdy-sys_1"  "Bdy-sys_10"  "Bdy-sys_11"  "Bdy-sys_12"  "Bdy-sys_13"
[3,] "Bdy-sys_1"  "Bdy-sys_10"  "Bdy-sys_11"  "Bdy-sys_12"  "Bdy-sys_13"
[4,] "Bdy-sys_1"  "Bdy-sys_10"  "Bdy-sys_11"  "Bdy-sys_12"  "Bdy-sys_13"
[5,] "Bdy-sys_1"  "Bdy-sys_10"  "Bdy-sys_11"  "Bdy-sys_12"  "Bdy-sys_13"
[6,] "Bdy-sys_1"  "Bdy-sys_10"  "Bdy-sys_11"  "Bdy-sys_12"  "Bdy-sys_13"

[1,] "Bdy-sys_14"  "Bdy-sys_15"  "Bdy-sys_2"  "Bdy-sys_3"  "Bdy-sys_4"  "Bdy-sys_5"
[2,] "Bdy-sys_14"  "Bdy-sys_15"  "Bdy-sys_2"  "Bdy-sys_3"  "Bdy-sys_4"  "Bdy-sys_5"
[3,] "Bdy-sys_14"  "Bdy-sys_15"  "Bdy-sys_2"  "Bdy-sys_3"  "Bdy-sys_4"  "Bdy-sys_5"
[4,] "Bdy-sys_14"  "Bdy-sys_15"  "Bdy-sys_2"  "Bdy-sys_3"  "Bdy-sys_4"  "Bdy-sys_5"
[5,] "Bdy-sys_14"  "Bdy-sys_15"  "Bdy-sys_2"  "Bdy-sys_3"  "Bdy-sys_4"  "Bdy-sys_5"
[6,] "Bdy-sys_14"  "Bdy-sys_15"  "Bdy-sys_2"  "Bdy-sys_3"  "Bdy-sys_4"  "Bdy-sys_5"

[1,] "Bdy-sys_6"  "Bdy-sys_7"  "Bdy-sys_8"  "Bdy-sys_9"
[2,] "Bdy-sys_6"  "Bdy-sys_7"  "Bdy-sys_8"  "Bdy-sys_9"
[3,] "Bdy-sys_6"  "Bdy-sys_7"  "Bdy-sys_8"  "Bdy-sys_9"
[4,] "Bdy-sys_6"  "Bdy-sys_7"  "Bdy-sys_8"  "Bdy-sys_9"
[5,] "Bdy-sys_6"  "Bdy-sys_7"  "Bdy-sys_8"  "Bdy-sys_9"
[6,] "Bdy-sys_6"  "Bdy-sys_7"  "Bdy-sys_8"  "Bdy-sys_9"

## End(Not run)
```

---

**c212.interim.1a.hier3**  
A Three-Level Hierarchical Body-system based Model for interim analysis without Point-Mass.
Description


Usage

c212.interim.1a.hier3(trial.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),  
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

trial.data A file or data frame containing the trial data. It must contain the columns _index (interval index), B (body-system), AE (adverse event). Group (1 - control, 2 treatment), Count (total number of events), Total (total number of participants in the trial arm).

sim_type The type of MCMC method to use for simulating from non-standard distributions. Allowed values are "MH" and "SLICE" for Metropolis-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 simulation 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; B: the body-system (if applicable); AE: the adverse event (if applicable); param: the simulation parameter; value: the overridden value; control: the overridden control value.

The function c212.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 c212.gen.initial.values can be used to generate a template for the list which can be updated by the user if required. The formats of the list elements are as follows:

- `gamma, theta`: dataframe with columns B, AE, chain, value
- `mu.gamma, mu.theta, sigma2.gamma, sigma2.theta`: dataframe with columns B, chain, value
- `mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0`: array of size chain.

level  The level of longitudinal dependency between the intervals. 0 - independent intervals, 1 - common interva body-system means, 2 - weak dependency.

hyper_params  The hyperparameters for the model. The default hyperparameters are those given in Berry and Berry 2004.

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, nIntervals, Intervals, nBodySys, maxBs, maxAEs, 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, gamma, theta, gamma_acc, theta_acc)
```
where

- `id` - a string identifying the version of the function
- `sim_type` - a string identifying the sampling method used for non-standard distributions, either "MH" or "SLICE"
- `chains` - the number of chains for which the simulation was run.
- `nIntervals` - the number of intervals in the simulation
- `Intervals` - an array. The intervals.
- `nBodySys` - the number of body-systems
- `maxBs` - the maximum number of body-systems in an interval
- `maxAEs` - the maximum number of AEs in a body-system
- `nAE` - an array. The number of AEs in each body-system.
- `AE` - an array of dimension `nBodySys, maxAEs`. The Adverse Events.
- `B` - an array. The body-systems.
- `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 samples of dimension `chains, iter - burnin`
- `mu.theta.0` - array of samples of dimension `chains, iter - burnin`
- `tau2.gamma.0` - array of samples of dimension `chains, iter - burnin`
- `tau2.theta.0` - array of samples of dimension `chains, iter - burnin`
- `mu.gamma` - array of samples of dimension `chains, nBodySys iter - burnin`
- `mu.theta` - array of samples of dimension `chains, nBodySys iter - burnin`
- `sigma2.gamma` - array of samples of dimension `chains, nBodySys iter - burnin`
- `sigma2.theta` - array of samples of dimension `chains, nBodySys iter - burnin`
- `gamma` - array of samples of dimension `chains, nBodySys, maxAEs, iter - burnin`
- `theta` - array of samples of dimension `chains, nBodySys, maxAEs, iter - burnin`
- `gamma_acc` - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used. An array of dimension `chains, nBodySys, maxAEs`
- `theta_acc` - the acceptance rate for the theta samples if a Metropolis-Hastings method is used. An array of dimension `chains, nBodySys, maxAEs`

**Note**

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

**Author(s)**

R. Carragher
Examples

```r
# Not run:
data(c212.trial.interval.data1)
raw = c212.interim.1a.hier3(c212.trial.interval.data1, level = 1)
raw$B

[1,] "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[2,] "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[3,] "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[4,] "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[5,] "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[6,] "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[1,] "Bdy-sys_14" "Bdy-sys_15" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5"
[2,] "Bdy-sys_14" "Bdy-sys_15" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5"
[3,] "Bdy-sys_14" "Bdy-sys_15" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5"
[4,] "Bdy-sys_14" "Bdy-sys_15" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5"
[5,] "Bdy-sys_14" "Bdy-sys_15" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5"
[6,] "Bdy-sys_14" "Bdy-sys_15" "Bdy-sys_2" "Bdy-sys_3" "Bdy-sys_4" "Bdy-sys_5"
[1,] "Bdy-sys_6" "Bdy-sys_7" "Bdy-sys_8" "Bdy-sys_9"
[2,] "Bdy-sys_6" "Bdy-sys_7" "Bdy-sys_8" "Bdy-sys_9"
[3,] "Bdy-sys_6" "Bdy-sys_7" "Bdy-sys_8" "Bdy-sys_9"
[4,] "Bdy-sys_6" "Bdy-sys_7" "Bdy-sys_8" "Bdy-sys_9"
[5,] "Bdy-sys_6" "Bdy-sys_7" "Bdy-sys_8" "Bdy-sys_9"
[6,] "Bdy-sys_6" "Bdy-sys_7" "Bdy-sys_8" "Bdy-sys_9"
```

## A Three-Level Hierarchical Body-system based Model for interim analysis with Point-Mass.

### Description


### Usage

```r
c212.interim.BB.hier2(trial.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,
hyperc.params = list(mu.gamma.0 = 0, tau2.gamma.0 = 10, mu.theta.0 = 0,
tau2.theta.0 = 10, alpha.gamma.3 = 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

trial.data A file or data frame containing the trial data. It must contain must contain the
columns B (body-system), AE (adverse event), Group (1 - control, 2 treatment),
Count (total number of events), Total (total number of participants).
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 distribu-
tions 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 simulation 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 param-
eters are the estimated width of the slice and a value limiting the search for the
next sample.
sim.params A data frame containing simulation parameters which override the global simula-
tion 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; B: the body-system (if applicable); AE: the adverse event (if applic-
able); param: the simulation parameter; value: the overridden value; control:
the overridden control value.
The function c212.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:
The function `c212.gen.initial.values` can be used to generate a template for the list which can be updated by the user if required. The formats of the list elements are as follows:

- **gamma, theta**: dataframe with columns \( B, AE, \text{chain}, \text{value} \)
- **mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi**: dataframe with columns \( B, \text{chain}, \text{value} \)

**level**

Allowed values are 0, 1. Respectively these indicate independent intervals, common body-system means across the intervals.

**hyper_params**

The hyperparameters for the model. The default hyperparameters are those given in Berry and Berry 2004.

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

- **pm.weights**
  Override global.pm.weight for specific adverse events.

- **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. The details of the complete conditional densities are given in Berry and Berry (2004).

### Value

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

- **list(id, theta_alg, sim_type, chains, nIntervals, Intervals, nBodySys, maxBs, maxAES, nAE, AE, B, burnin, iter, monitor, mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi, gamma, theta, gamma_acc, theta_acc)**

where

- **id** - a string identifying the version of the function
- **theta_alg** - an string identifying the algorithm used to sample theta
- **sim_type** - an string identifying the sampling method used for non-standard distributions, either "MH" or "SLICE"
- **chains** - the number of chains for which the simulation was run
- **nIntervals** - the number of intervals in the simulation
- **Intervals** - an array. The intervals.
- **nBodySys** - the number of body-systems
maxBs - the maximum number of body-systems in an interval
maxAEs - the maximum number of AEs in a body-system
nAE - an array. The number of AEs in each body-system.
AE - an array of dimension nBodySys, maxAEs. The Adverse Events.
B - an array. The body-systems.
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 samples of dimension chains, nBodySys iter - burnin
mu.theta - array of samples of dimension chains, nBodySys iter - burnin
sigma2.gamma - array of samples of dimension chains, nBodySys iter - burnin
sigma2.theta - array of samples of dimension chains, nBodySys iter - burnin
pi - array of samples of dimension chains, nBodySys iter - burnin
gamma - array of samples of dimension chains, nBodySys, maxAEs, iter - burnin
theta - array of samples of dimension chains, nBodySys, maxAEs, iter - burnin
gamma_acc - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used. An array of dimension chains, nBodySys, maxAEs
theta_acc - the acceptance rate for the theta samples. An array of dimension chains, nBodySys, maxAEs

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

Author(s)
R. Carragher

Examples

```r
## Not run:
data(c212.interim.1a.hier2)
raw = c212.interim.1a.hier2(c212.interim.1a.hier2(1), level = 1)
```

raw$B

```
[1,]  "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[2,]  "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[3,]  "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[4,]  "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[5,]  "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
[6,]  "Bdy-sys_1" "Bdy-sys_10" "Bdy-sys_11" "Bdy-sys_12" "Bdy-sys_13"
```
A Three-Level Hierarchical Body-system based Model for interim analysis with Point-Mass.

Description


Usage

c212.interim.BB.hier3(trial.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), 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 = 3, alpha.theta.0.0 = 3, alpha.gamma = 3, beta.gamma = 3, alpha.theta = 3, beta.theta = 3, beta.thetav = 1, level = 1)
library(c212)

# Example parameters
lambda.alpha = 1.0, lambda.beta = 1.0,
global.pm.weight = 0.5,
pm.weights = NULL,
adapt_phase = 1, memory_model = "HIGH"

**Arguments**

- **trial.data**
  A file or data frame containing the trial data. It must contain
  must contain the columns `B` (body-system), `AE` (adverse event), `Group` (1 - control, 2 treatment),
  `Count` (total number of events), `Total` (total number of participants).

- **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 simulation 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 param-
  eters 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 simula-
  tion 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; B: the body-system (if applicable); AE: the adverse event (if applic-
  able); param: the simulation parameter; value: the overridden value; control:
  the overridden control value.

  The function `c212.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 `c212.gen.initial.values` can be used to generate a template for the
  list which can be updated by the user if required. The formats of the list elements
  are as follows:
gamma, theta: dataframe with columns B, AE, chain, value
mu.gamma, mu.theta, sigma2.gamma, sigma2.theta, pi: dataframe with columns B, chain, value
mu.gamma.0, mu.theta.0, tau2.gamma.0, tau2.theta.0, alpha.pi, beta.pi: array of size chain.

level Allowed values are 0, 1, 2. Respectively these indicate independent intervals, common body-system means across the intervals and weak relationships between the intervals.

hyper_params The hyperparameters for the model. The default hyperparameters are those given in Berry and Berry 2004.

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

pm.weights Override global.pm.weight for specific adverse events.

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. The details of the complete conditional densities are given in Berry and Berry (2004).

Value

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

`list(id, theta_alg, sim_type, chains, nIntervals, Intervals, nBodySys, maxBs, maxAEs, 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 versions of the function

`theta_alg` - an string identifying the algorithm used to sample theta

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

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

`nIntervals` - the number of intervals in the simulation

`Intervals` - an array. The intervals.

`nBodySys` - the number of body-systems

`maxBs` - the maximum number of body-systems in an interval

`maxAEs` - the maximum number of AEs in a body-system

`nAE` - an array. The number of AEs in each body-system.
AE - an array of dimension \( nBodySys, maxAEs \). The Adverse Events.
\( B \) - an array. The body-systems.
\( burnin \) - 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 samples of dimension \( chains, iter - burnin \)
\( mu.theta.0 \) - array of samples of dimension \( chains, iter - burnin \)
\( tau2.gamma.0 \) - array of samples of dimension \( chains, iter - burnin \)
\( tau2.theta.0 \) - array of samples of dimension \( chains, iter - burnin \)
\( mu.gamma \) - array of samples of dimension \( chains, nBodySys, iter - burnin \)
\( mu.theta \) - array of samples of dimension \( chains, nBodySys, iter - burnin \)
\( sigma2.gamma \) - array of samples of dimension \( chains, nBodySys, iter - burnin \)
\( sigma2.theta \) - array of samples of dimension \( chains, nBodySys, iter - burnin \)
\( pi \) - array of samples of dimension \( chains, nBodySys, iter - burnin \)
\( alpha.pi \) - array of samples of dimension \( chains, iter - burnin \)
\( beta.pi \) - array of samples of dimension \( chains, iter - burnin \)
\( gamma_acc \) - the acceptance rate for the gamma samples if a Metropolis-Hastings method is used.
An array of dimension \( chains, nBodySys, maxAEs \)
\( theta_acc \) - the acceptance rate for the theta samples. An array of dimension \( chains, nBodySys, maxAEs \)

Note

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

Author(s)

R. Carragher

Examples

```r
## Not run:
data(c212.trial.interval.data1)
raw = c212.interim.1a.hier3(c212.trial.interval.data1, level = 1)

raw$B
1 0.8  1  2  3  4
2  5  6  7  8  9
```

**c212.interim.MLE**

---

### Poisson Maximum Likelihood Estimator

#### Description

Calculate the Poisson Maximum Likelihood Estimator for interim analysis data.

#### Usage

```r
c212.interim.MLE(trial.data)
```

#### Arguments

- **trial.data**
  
  A file or data frame containing the trial data. It must contain the columns `I_index` (interval index), `B` (body-system), `AE` (adverse event), `Group` (1 - control, 2 treatment), `Count` (total number of events), `Total` (total number of participants in the trial arm).

#### Value

The maximum likelihood estimators and summary statistics.

#### Author(s)

R. Carragher
Examples

```r
## Not run:
data(c212.trial.interval.data1)
raw = c212.interim.MLE(c212.trial.interval.data1)

## End(Not run)
```

Implementation of the least-slope estimator estimator (LSL) for the proportion of true null hypotheses.

Description

The least-slope estimator estimator (LSL) is one of a number of estimators of the proportion of true null hypotheses. This implementation assumes a grouped structure for the data.

Usage

```r
c212.LSL(trial.data)
```

Arguments

Arguements
trial.data Data frame containing the p-values for the hypotheses being tested. The data must contain the following columns: $B$: the index or name of the groupings; $p$: the p-values of the hypotheses.

Value

An estimate of the proportion of true null hypotheses.

Note

The implementation is that described in Hu, J. X. and Zhao, H. and Zhou, H. H. (2010).

Author(s)

R. Carragher<raymond.carragher@strath.ac.uk>

References


Generate a template for choosing which samples to monitor.

Description

This function generates 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 reduces the overall memory footprint.

Usage

c212.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

## Not run:
c212.monitor.samples("1a", hier = 3)
## End(Not run)

Unadjusted test of multiple hypotheses.

Description

Unadjusted test of multiple hypotheses.

Usage

c212.NOADJ(trial.data, alpha = 0.05)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>trial.data</td>
<td>File or data frame containing the p-values for the hypotheses being tested. The data must include a column called ( p ) which contains the p-values of the hypotheses.</td>
</tr>
<tr>
<td>alpha</td>
<td>The level for FDR control. E.g. 0.05.</td>
</tr>
</tbody>
</table>

Value

The subset of hypotheses in `file` or `trial.data` deemed significant at level `alpha`.

Note

No check is made for duplicate rows in the input file or data frame.

Author(s)

R. Carragher

Examples

```r
trial.data <- data.frame(B = c(1, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4),
                        j = c(1, 1, 2, 3, 4, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5),
                        p = c(0.135005, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023, 0.011, 0.023000, 0.016, 0.0109, 0.559111, 0.751986, 0.300339, 0.837154, 0.325822))
c212.NOADJ(trial.data, alpha=0.05)
```

```
## Not run:
B j  AE  p
1 2 2 AE3 0.0010
2 3 2 AE7 0.0013
3 3 3 AE8 0.0023
4 2 3 AE4 0.0050
5 2 1 AE2 0.0100
6 3 7 AE12 0.0109
7 3 4 AE9 0.0110
8 3 6 AE11 0.0160
9 3 1 AE6 0.0200
10 3 5 AE10 0.0230

## End(Not run)
```
**Plot Adverse Event Incidence Data**

**Description**

This function plots a graph of the total adverse event incidence counts by body-system and by individual adverse event for end of trial data.

**Usage**

```r
C212.plot.eot.data(trial.data, legend = TRUE, interactive = FALSE,
cex = 0.5)
```

**Arguments**

- `trial.data`: A file or data frame containing the trial data. The data frame must contain the columns `B` (body-system), `AE` (adverse event), `Group` (1 - control, 2 - treatment), `Count` (total number of events), `Total` (total number of participants).
- `legend`: Boolean. If TRUE print a legend.
- `interactive`: Boolean. If TRUE allow the user to identify individual adverse events on the individual adverse events graph.
- `cex`: Font size of the labels on the Adverse Event counts graph.

**Details**

Two graphs are displayed on the same panel. The top graph is of AE Incidence Counts by Body-System. The lower graphs is of the Individual AE Incidence Counts.

**Value**

Nothing is returned.

**Note**

The legend may obscure a portion of the graph. In this case the legend may be suppressed by choosing `legend = FALSE` when calling the function.

**Author(s)**

R. Carragher

**Examples**

```r
## Not run:
data(c212.trial.data)
c212.plot.eot.data(c212.trial.data)

## End(Not run)
```
c212.plot.interim.data rd

Plot Adverse Event Count Data for a Body-system by Interval

Description

Plot adverse event interval data for a body-system.

Usage

c212.plot.interim.data(trial.data, body_sys, cex = 0.8, title = NULL)

Arguments

- **trial.data**: A file or data frame containing the trial data. The data frame must contain the columns `I_index` (interval), `B` (body-system), `AE` (adverse event), `Group` (1 - control, 2 treatment), `Count` (total number of events), `Exposure` (total time of participants speant in the interval).

- **body_sys**: The body-system for which to plot the events.

- **cex**: Font size of the labels on the Adverse Event counts graph.

- **title**: Main title of the graph.

Details

This function plots a graph of the count of adverse events which have occurred in an interval for a particular body-system by interval.

Value

Nothing is returned.

Author(s)

R. Carragher

Examples

```r
## Not run:
data(c212.trial.interval.data1)
c212.plot.interim.data(c212.trial.interval.data1, "Bdy-sys_3")
## End(Not run)
```
c212.plot.samples  

Plot Posterior Distribution

Description

This function plots a graph of the sampled posterior distribution.

Usage

c212.plot.samples(samples, title)

Arguments

- `samples`: An array of samples indexed by `chain`.
- `title`: The graph title.

Details

Two graphs are displayed on the same panel. The left graph is the traceplot of the chains. The right graph is a plot of the distribution.

Value

Nothing is returned.

Author(s)

R. Carragher

Examples

```r
## Not run:
data(c212.trial.data)
raw = c212.1a(c212.trial.data)
sample = raw$theta[,2,2]
c212.plot.samples(sample, sprintf("%s: %s %s", "theta", raw$B[2], raw$AE[2]))

## End(Not run)
```
c212.pointmass.weights

Generate a template for the point-mass weightings.

Description

This function generates a template for weights for the proposal distribution used to sample \( \theta \) variables in models which use a point-mass.

Usage

\[
c212\text{-}\text{pointmass.weights}(\text{trial.data})
\]

Arguments

- \text{trial.data} A file or data frame containing the trial data, either for end of trial or interim analysis.

Value

A dataframe containing the weightings template for each Body-system, adverse event and, if required, interval.

Author(s)

R. Carragher

Examples

\[
\begin{align*}
\text{## Not run:} \\
\text{data(c212\_trial\_data)} \\
\text{c212\_pointmass\_weights(c212\_trial\_data)} \\
\text{## End(Not run)}
\end{align*}
\]

c212.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

\[
c212\text{-}\text{print.convergence.summary}(\text{conv})
\]
Arguments
conv The output from a call to \texttt{c212.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
\begin{verbatim}
## Not run:
data(c212.trial.data)
raw = c212.BB(c212.trial.data)
conv = c212.check.convergence(rm)
c212.print.convergence.summary(conv)

## End(Not run)
\end{verbatim}

---

\texttt{c212.print.summary.stats}

\textit{Print the Summary Statistics of Posterior Distributions}

Description
The function prints the variable names, the mean, the 95 MCMC standard error for the simulated sample.

Usage
c212.print.summary.stats(summ)

Arguments
summ The output from a call to \texttt{c212.summary.stats}

Value
Nothing
Author(s)
R. Carragher

Examples

```r
## Not run:
summ = c212.summary.stats(rm)
c212.print.summary.stats(summ)

## End(Not run)
```

**c212.ptheta**

*Reports the posterior probability that theta (the increase in the log-odds) is greater than zero for each Adverse Event*

**Description**

This function reports the posterior probability that theta is positive, i.e. that there is an increase in log odds of an adverse event being associated with treatment.

**Usage**

```r
c212.ptheta(raw)
```

**Arguments**

- `raw` The output from a call to `c212.BB`.

**Value**

A data frame containing the columns: `interval` if analysing interim data, `B`: body system, `AE`: adverse event and `ptheta`, the posterior probability that theta is positive.

**Author(s)**

R. Carragher

**Examples**
## Not run:
data(c212.trial.data)
raw = c212.BB(c212.trial.data)

## End(Not run)

## Not run:
p = c212.BB.ptheta(rm)

head(p)

<table>
<thead>
<tr>
<th>B</th>
<th>AE</th>
<th>ptheta</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adv-Ev_1</td>
<td>0.2560500</td>
</tr>
<tr>
<td>2</td>
<td>Adv-Ev_2</td>
<td>0.9426417</td>
</tr>
<tr>
<td>3</td>
<td>Adv-Ev_3</td>
<td>0.8751500</td>
</tr>
<tr>
<td>4</td>
<td>Adv-Ev_4</td>
<td>0.1154917</td>
</tr>
<tr>
<td>5</td>
<td>Adv-Ev_5</td>
<td>0.2317417</td>
</tr>
<tr>
<td>6</td>
<td>Adv-Ev_6</td>
<td>1.0000000</td>
</tr>
</tbody>
</table>

## End(Not run)
c212.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. c212.BB).

Usage
```
c212.sim.control.params(trial.data, model = "1a")
```

Arguments
- `trial.data`: A file or data frame containing the trial data, either for end of trial or interim analysis.
- `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
```
## Not run:
data(c212.trial.data)
c212.sim.control.params(c212.trial.data)
## End(Not run)
```

Implementation of Subset Benjamini-Hochberg for False Discover Rate control

Description
The Subset Benjamini-Hochberg allows for the use of subsets to allow the extension of the Benjamini-Hochberg procedure to types of non-positively dependent regression statistics.
Usage

c212.ssBH(trial.data, alpha = 0.05)

Arguments

trial.data File or data frame containing the p-values for the hypotheses being tested. The data must contain the following columns: B: the index of the groupings; p: the p-values of the hypotheses.

alpha The level for FDR control. E.g. 0.05.

Value

The subset of hypotheses in file or trial.data deemed significant by the Subset Benjamini-Hochberg process.

Note

This process is at most as powerful as the Benjamini-Hochberg procedure. The subsets do not have to be disjoint.

Author(s)

R. Carragher

References


Examples

trial.data <- data.frame(B = c(1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4),
                         j = c(1, 2, 2, 3, 4, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5),
                                "AE12", "AE13", "AE14", "AE15", "AE16", "AE17"),
                         p = c(0.153505, 0.010000, 0.001000, 0.005000, 0.153501, 0.020000, 0.0013, 0.0023,
                               0.011, 0.023000, 0.016, 0.0109, 0.559111, 0.751906, 0.380339, 0.837154, 0.325802))

c212.ssBH(trial.data, alpha=0.05)

## Not run:
B j AE p
1 2 2 AE3 0.0010
2 2 3 AE4 0.0050
3 3 2 AE7 0.0013
4 3 3 AE8 0.0023
5 3 7 AE12 0.0109
6 3 4 AE9 0.0110

## End(Not run)
**c212.summary.stats**

Summary Statistics for the Posterior Distributions in the model.

**Description**

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

**Usage**

```r
c212.summary.stats(raw)
```

**Arguments**

- `raw` The output from a model simulation (e.g. `c212.BB`).

**Details**

The function reports the mean, upper and lower bounds of the 95 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`, `hpi_lower`, `hpi_upper`, `SD` and `SE`. For the simulation return by `c212.1a` the output is as follows:

```r
list(theta.summary, gamma.summary, mu.gamma.summary, 
     mu.theta.summary = mu.theta_summ, 
     sigma2.gamma.summary, sigma2.theta.summary, 
     mu.gamma.0.summary, mu.theta.0.summary, 
     tau2.gamma.0.summary, tau2.theta.0.summary)
```

Additional columns which may be used to indentify the individual variables are `B`, the body-system, and `AE`, the Adverse Event and `interval`.

**Note**

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

**Author(s)**

R. Carragher
Examples
## Not run:
data(c212.trial.data)
raw = c212.BB(c212.trial.data)
summ = c212.summary.stats(raw)

## End(Not run)

c212.trial.data  End of Trial Data Clinical Data for Adverse Event Indicence

Description
This data set contains the counts of adverse event incidence for the trial.

Usage
c212.trial.data

Format
A dataframe with columns B - body-system, AE - adverse event, Group - 1 for control, 2 for treatment, Count - total adverse event incidence, Total - total patients on the trial arm. The dataframe contains 90 observations.

c212.trial.interval.data1  Interim analysis trial data.

Description
This data set contains count of the adverse events over the first two intervals of a clinical trial.

Usage
c212.trial.interval.data1

Format
A dataframe with columns interval, I_index - the interval order, B - body-system, AE - adverse event, Group - 1 for control, 2 for treatment, Count - total adverse events that occurred in the interval, Exposure - the total time all at risk subjects spent in the interval. The dataframe contains 1860 observations.
**c212.trial.interval.data2**

*Interim analysis trial data.*

**Description**

This data set contains count of the adverse events over all intervals of a clinical trial.

**Usage**

c212.trial.interval.data2

**Format**

A dataframe with columns `interval`, `I_index` - the interval order, `B` - body-system, `AE` - adverse event, `Group` - 1 for control, 2 for treatment, `Count` - total adverse events that occurred in the interval, `Exposure` - the total time all at risk subjects spent in the interval. The dataframe contains 3100 observations.

---

**c212.TST**

*Implementation of the two-stage estimator (TST) for the proportion of true null hypotheses.*

**Description**

The two-stage estimator (TST) is one of a number of estimators of the proportion of true null hypotheses. It uses the Benjamini-Hochberg procedure at a reduced level to make the estimate. This implementation assumes a grouped structure for the data.

**Usage**

c212.TST(trial.data, alpha)

**Arguments**

- **trial.data**  
  Data frame containing the p-values for the hypotheses being tested. The data must contain the following columns: `B`: the index or name of the groupings; `p`: the p-values of the hypotheses.

- **alpha**  
  The level for FDR control. E.g. 0.05.

**Value**

An estimate of the proportion of true null hypotheses.
Note

The implementation is that described in Hu, J. X. and Zhao, H. and Zhou, H. H. (2010).

Author(s)

R. Carragher<raymond.carragher@strath.ac.uk>

References


Index

*Topic Adverse Event
  c212.package, 2
*Topic BH adjusted p-values
  c212.BH.adjust.pvals, 14
*Topic BH
  c212.BH, 12
*Topic Bayesian Hierarchy
  c212.package, 2
*Topic Bayesian
  c212.1a, 4
c212.BB, 7
c212.convergence.diag, 18
c212.gen.initial.values, 24
c212.interim.1a.hier2, 25
c212.interim.1a.hier3, 28
c212.interim.BB.hier2, 32
c212.interim.BB.hier3, 36
c212.monitor.samples, 42
c212.plot.eot.data, 44
c212.plot.interim.data.rd, 45
c212.plot.samples, 46
c212.pointmass.weights, 47
c212.print.convergence.summary, 47
c212.print.summary.stats, 48
c212.ptheta, 49
c212.sim.control.params, 51
c212.summary.stats, 53
*Topic Benjamini-Hochberg adjusted p-values
  c212.BH.adjust.pvals, 14
*Topic Benjamini-Hochberg
  c212.BH, 12
*Topic Berry and Berry
  c212.1a, 4
c212.BB, 7
c212.convergence.diag, 18
c212.gen.initial.values, 24
c212.interim.1a.hier2, 25
c212.interim.1a.hier3, 28
c212.interim.BB.hier2, 32
c212.interim.BB.hier3, 36
c212.monitor.samples, 42
c212.plot.eot.data, 44
c212.plot.interim.data.rd, 45
c212.plot.samples, 46
c212.pointmass.weights, 47
c212.print.convergence.summary, 47
c212.print.summary.stats, 48
c212.ptheta, 49
c212.sim.control.params, 51
c212.summary.stats, 53
*Topic False Discovery Rate
  c212.package, 2
*Topic False Discovery Rate
  c212.BONF, 17
*Topic Fisher
  c212.fisher.test, 21
*Topic GBH
  c212.package, 2
*Topic Gelman-Rubin
  c212.convergence.diag, 18
c212.print.convergence.summary, 47

*Topic **Group Benjamini-Hochberg**
c212-package, 2

*Topic **Hierarchy**
c212.1a, 4
c212.BB, 7
c212.convergence.diag, 18

c212.interim.1a.hier2, 25
c212.interim.1a.hier3, 28
c212.interim.BB.hier2, 32

c212.interim.BB.hier3, 36
c212.monitor.samples, 42

c212.plot.eot.data, 44
c212.plot.interim.data.rd, 45

c212.plot.samples, 46

c212.pointmass.weights, 47

c212.print.convergence.summary, 47

c212.print.summary.stats, 48

c212.ptheta, 49

c212.sim.control.params, 51

c212.summary.stats, 53

*Topic **Hypothesis**
c212.bin.test, 16

*Topic **Interim analysis**
c212-package, 2

*Topic **Point-mass**
c212.BB, 7
c212.interim.BB.hier2, 32

c212.interim.BB.hier3, 36

*Topic **Poisson MLE**
c212.interim.MLE, 40

*Topic **Proportion**
c212.bin.test, 16

*Topic **Subset Benjamin-Hochberg**
c212-package, 2

*Topic **Subset Benjamini-Hochberg**
c212.ssBH, 51

*Topic **System organ class**
c212-package, 2

*Topic **Test**
c212.bin.test, 16
c212.fisher.test, 21

*Topic **Unadjusted testing**
c212.NOADJ, 42

*Topic **c212-package**
c212-package, 2

*Topic **c212.1a**