Package: BATSS (via r-universe)

October 6, 2026

Title Bayesian Adaptive Trial Simulator Software (BATSS) for Generalised Linear Models

Version 1.2.1

Description Defines operating characteristics of Bayesian Adaptive Trials considering a generalised linear model response via Monte Carlo simulations of Bayesian GLM fitted via integrated Laplace approximations (INLA).

URL https://batss-stable.github.io/BATSS/

License GPL-2

Encoding UTF-8

Suggests INLA

Imports parallel, methods, stats, grDevices, abind, plyr, rlang, R.utils, cli, sm

Additional_repositories https://inla.r-inla-download.org/R/testing

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Repository https://batss-stable.r-universe.dev

Date/Publication 2026-08-06 14:41:59 UTC

RemoteUrl https://github.com/batss-stable/batss

RemoteRef HEAD

RemoteSha a497574bdd35d76e4f50a6e12d7ca5e6b0da692a

Contents

alloc.balanced 2
alloc.simple 3
BATSS 3
batss.combine 4
batss.glm 5
eff.arm.infofract 9

eff.arm.simple 10
eff.trial.all 10
eff.trial.any 11
fut.arm.simple 11
fut.trial.all 12
plot.batss 12
print.batss 14
print.summary.batss 15
RAR.optimal 16
RAR.trippa 16
summary.batss 17
violin.density 18

Index **19**

`alloc.balanced` *Balanced allocation function*

Description

alloc.balanced first allocates the largest possible number of units to the different groups given their exact target probabilities and then assigns randomly the remaining units to the different groups according to multinomial draws. This method leads to observed allocation probabilities matching the target ones when m*prob is an integer for each group and to observed allocation probabilities (on average) closer to the target ones compared to alloc.simple.

Usage

```
alloc.balanced(m, prob)
```

Arguments

`m`	the 'BATSS' ingredient '`m`', a scalar corresponding to the number of participants to be allocated.
`prob`	the 'BATSS' ingredient '`prob`', a named vector of allocation ratios or probabilities.

Value

alloc.balanced returns an object of class factor of length '`m`' with levels matching the names of the vector '`prob`'.

See Also

`alloc.simple()`, another group allocation function.

Examples

```
alloc.balanced(100, prob = c(A=.4,B=.6))
table(alloc.balanced(100, prob = c(A=.4,B=.6)))
table(alloc.balanced(100, prob = c(A=.4,B=.6)))
```

`alloc.simple` *Simple allocation function*

Description

alloc.simple independently randomises each unit to a group (i.e., flips a coin for each unit) so that the observed allocation probabilities may be far from the target ones. This strategy is often considered to be a poor choice.

Usage

```
alloc.simple(m, prob)
```

Arguments

`m`	the 'BATSS' ingredient '`m`', a scalar corresponding to the number of participants to be allocated.
`prob`	the 'BATSS' ingredient '`prob`', a named vector of allocation ratios or probabilities.

Value

alloc.simple returns an object of class factor of length '`m`' with levels matching the names of the vector '`prob`'.

See Also

`alloc.balanced()`, another group allocation function.

Examples

```
alloc.simple(100, prob = c(A=.4,B=.6))
table(alloc.simple(100, prob = c(A=.4,B=.6)))
table(alloc.simple(100, prob = c(A=.4,B=.6)))
```

`BATSS` *BATSS*

Description

BATSS

`batss.combine` *Combines outputs generated by batss.glm*

Description

Combines different evaluations of batss.glm considering the same trial design but different sets of seeds. This function is useful when the evaluation of Monte Carlo samples generated by different seeds was split in sets computed by different nodes/cpus. The output of this function is of class 'batss' meaning that the usual generic functions (print, summary, plot) can be used.

Usage

```
batss.combine(paths, force = FALSE)
```

Arguments

`paths`	Vector indicating the paths to the rdata files containing the outputs of the function batss.glm considering the same trial design but different set of seeds. This requires the argument '`extended`' of the function batss.glm to be > 0.
`force`	a logical with default `force=FALSE`. Among other checks, batss.glm controls that the calls of the Monte Carlo trials to be combined are identical and stops if they are not (Note that this check is not bullet proof: such a check, for example, would be able to note that two sets of Monte Carlo trials used a `eff.arm` function named the same way and considered the same optional parameters but would be blind to the fact that they could correspond to two *different* functions). `force=TRUE` forces batss.glm to ignore this check. This could be useful if the calls differ due to the `batss` objects to be combined being generated using different versions of batss.glm.

Value

an object of class 'batss'. Refer to the section 'Value' in batss.glm for details about this object structure.

See Also

`batss.glm()`, the function allowing to simulate Bayesian adaptive trials with GLM endpoint for different seeds.

batss.glm *Bayesian adaptive trial simulations for generalised linear models*

Description

Simulation of Bayesian adaptive trials with GLM endpoint using Integrated Nested Laplace Approximation (INLA).

Usage

```
batss.glm(
  model,
  var,
  var.control = NULL,
  family = "gaussian",
  link = "identity",
  beta,
  which,
  alternative = "greater",
  R = 10000,
  N,
  interim,
  prob0,
  delta.eff = 0,
  delta.fut = delta.eff,
  delta.RAR = 0,
  eff.arm,
  eff.arm.control = NULL,
  eff.trial = NULL,
  eff.trial.control = NULL,
  fut.arm,
  fut.arm.control = NULL,
  fut.trial = NULL,
  fut.trial.control = NULL,
  RAR = NULL,
  RAR.control = NULL,
  H0 = TRUE,
  computation = "parallel",
  mc.cores = getOption("mc.cores", 3L),
  extended = 0,
  ...
)
```

Arguments

model	an object of class 'formula' indicating a symbolic description of the model to be fitted (as in the lm and glm functions).

`var` A list. Each entry corresponds to a variable described under '`model`' and indicates the name of a function allowing to generate variates (like rnorm and rexp, for example). The list names have to match the variable names unded in '`model`' and its first element should correspond to the model outcome. The grouping variable corresponding to the target parameters has to be of class 'factor' with levels corresponding to the names indicated in argument `prob0` (see below).

`var.control` An optional list of control parameters for the functions indicated in '`var`'. The names of the list items need to correspond to the names used in '`var`'. Each element is another list with names of the elements corresponding to the parameter names of the functions specified in '`var`'.

`family` A character string indicating the name of the conditional distribution as described in the package INLA (check inla.list.models). Default set to '`gaussian`'.

`link` A character string describing the link function to be used in the model to relate the outcome to the set of predictors: 'identity', 'log', 'logit', 'probit', 'robit', 'cauchit', 'loglog' and 'cloglog' are the currently available options. Default set to 'identity'.

`beta` A numerical vector of parameter values for the linear predictor. Its length has to match the number of column of the $\mathbf{X}$ matrix induced by the formula indicated under '`model`' (check model.matrix).

`which` A numerical vector indicating the position of the target `beta` parameters.

`alternative` A vector of strings providing the one-sided direction of the alternative hypothesis corresponding to each target parameter indicated under '`which`' (in the same order). Possibilities are 'greater' (default) or 'less'. If the vector is of length 1, the same direction will be used for all target parameter tests.

`R` a vector of natural numbers to be used as seeds (check set.seed) for the different Monte Carlo trials (the vector length will thus correspond to the number of Monte Carlo trials). When `R` is a scalar, seeds `1` to `R` are used, where `R` corresponds to the number of Monte Carlo trials.

`N` A scalar indicating the maximum sample size.

`interim` A list of parameters related to interim analyses. Currently, only '`recruited`' is available. It consists in a vector of integers indicating the number of completed observations at each look, last excluded, in increasing order. Setting `interim = NA` specifies a fixed (non-adaptive) design with a single look at the maximum sample size `N`.

`prob0` A named vector with initial allocation probabilities. Names need to correspond to the levels of the grouping variable. If `RAR = NULL`, these probabilities/ratios will be used throughout (fixed allocation probabilities).

`delta.eff` A vector (of length equal to the number of looks (i.e., number of interims + 1)) of clinically meaningful treatment effect values (on the linear predictor scale) to be used to define the efficacy-related posterior probabilities for each target parameter at each look. If a scalar is provided, the same value is used at each look. The default is `delta.eff = 0`.

`delta.fut` A vector (of length equal to the number of looks (i.e., number of interims + 1)) of clinically meaningful treatment effect values (on the linear predictor scale) to be used to define the futility-related posterior probabilities for each target

parameter at each look. If a scalar is provided, the same value is used at each look. The default is `delta.fut = delta.eff`.

`delta.RAR` A vector (of length equal to the number of looks (i.e., number of interims + 1)) of clinically meaningful treatment effect values (on the linear predictor scale) to be used to define the RAR-related posterior probabilities for each target parameter at each look. If a scalar is provided, the same value is used at each interim analysis. The default is `delta.RAR = 0`. Note that, when a vector is provided, its last value is ignored as no randomisation is made at the last look.

`eff.arm` A function defining if efficacy has been achieved at a given look given the information available at that stage a given target parameter. The output of this function must be a logical (of length 1). Arguments of this function will typically consider 'BATSS' ingredients. Check eff.arm.simple and eff.arm.infofract for examples.

`eff.arm.control`
An optional list of parameters for the function indicated in '`eff.arm`'.

`eff.trial` A function defining if the trial can be stopped for efficacy given the output of the function indicated in '`eff.arm`'. The output of this function must be a logical of length one. Arguments of this function will typically only consider the 'BATSS' ingredient `eff.target`. Check eff.trial.all and eff.trial.any for examples. When `eff.trial = NULL` (default), the trial stops for efficacy when *all* target parameters are found to be effective (like in eff.trial.all).

`eff.trial.control`
An optional list of parameters for the function indicated in '`eff.trial`'.

`fut.arm` A function defining if futility has been achieved at a given look given the information available at that stage for each target parameter. The output of this function must be a logical (of length 1). Arguments of this function will typically consider 'BATSS' ingredients. Check fut.arm.simple to see an example of such a function.

`fut.arm.control`
An optional list of parameters for the function indicated in '`fut.arm`'.

`fut.trial` A function defining if the trial can be stopped for futility given the output of the function indicated in '`fut.arm`'. The output of this function must be a logical of length one. Arguments of this function will typically only consider the 'BATSS' ingredient `fut.target`. Check fut.trial.all for an example of such a function. When `fut.trial = NULL` (default), the trial stops for futility when *all* target parameters are found to be futile (like in fut.trial.all).

`fut.trial.control`
An optional list of parameters for the function indicated in '`fut.trial`'.

`RAR` A function defining the response-adaptive randomisation probabilities of each group - reference group included - with the same group names and ordering as used in '`prob0`'. Arguments of this function will typically consider 'BATSS' ingredients. Check RAR.trippa and RAR.optimal for examples. If `RAR = NULL` (default), the probabilities/ratios indicated under `prob0` will be used throughout (fixed allocation probabilities).

`RAR.control` An optional list of control parameters for the function provided in '`RAR`'.

H0	A logical indicating whether the simulation should also consider the case with all target parameters set to 0 to check the probability of rejecting the hypothesis that the target parameter value is equal to 0 individually (pairwise type I error) or globally (family-wise error rate). Default set to H0=TRUE.
computation	A character string indicating how the computation should be performed. Possibilities are 'parallel' or 'sequential' with default computation="parallel" meaning that the computation is split between mc.cores.
mc.cores	An integer indicating the number of CPUs to be used when computation="parallel" (Default to 3 if no global 'mc.cores' global option is available via getOption).
extended	an integer indicating the type of results to be returned. 0 (default) provides summary statistics, 1 adds the results of each Monte Carlo trial and 2 additionally returns each Monte Carlo dataset. batss.combine requires extended > 0 as the function needs to merge results of different sets of seeds.
...	Additional arguments to control fitting in inla.

Value

The function batss.glm returns an S3 object of class 'batss' with available print/summary/plot functions

- beta - A data frame providing information related to the beta parameter vector, like parameter names and values, for example.
- look - A data frame providing information related to looks, like sample size of a given interim (m) and cumulative sample size at a given interim (n), for example.
- par - A list providing different information, like the used seeds (seed) and the groups (group), for example.
- H1 - A list providing trial results under the alternative, like the estimates per target parameter when the corresponding arm was stopped (estimate), the efficacy and futility probabilites per target parameter and overall (target, efficacy and futility), the sample size per group and trial (sample), the probabilities associated to each combination of efficacy and futility per group (scenario), the detailed results per trial (trial), for example.
- H0 - A list providing trial results under the global null hypothesis (same structure as H1).
- call - The matched call.
- type - The type of 'BATSS' analysis (only 'glm' is currently available).

See Also

summary.batss and plot.batss for detailed summaries and plots, and batss.combine to combine different evaluations of batss.glm considering the same trial design but different sets of seeds (useful for cluster computation).

Examples

```
# Example:
# * Gaussian conditional distribution with sigma = 5
# * 3 groups with group means 'C' = 1 (ref), 'T1' = 2, 'T2' = 3,
#     where higher means correspond to better outcomes
```

```
# * 5 interim analyses occurring when n = 100, 120, 140, 160, and 180
# * fixed and equal allocation probabilities per arm (i.e., no RAR)
# * max sample size = 200
# * efficacy stop per arm when the prob of the corresponding parameter
#     being greater than 0 is greater than 0.975 (?eff.arm.simple)
# * futility stop per arm when the prob of the corresponding parameter
#     being greater than 0 is smaller than 0.05 (?fut.arm.simple)
# * trial stop once all arms have stopped (?eff.trial.all and ?fut.trial.all)
#     or the max sample size was reached

sim = batss.glm(model           = y ~ group,
                var             = list(y     = rnorm,
                                       group = alloc.balanced),
                var.control     = list(y = list(sd = 5)),
                beta            = c(1, 1, 2),
                which           = c(2:3),
                alternative     = "greater",
                R               = 20,
                N               = 200,
                interim         = list(recruited = seq(100, 180, 20)),
                prob0           = c(C = 1/3, T1 = 1/3, T2 = 1/3),
                eff.arm         = eff.arm.simple,
                eff.arm.control = list(b = 0.975),
                fut.arm         = fut.arm.simple,
                fut.arm.control = list(b = 0.05),
                computation     = "parallel",
                H0              = TRUE,
                mc.cores        = 2)# better: parallel::detectCores()-1
```

`eff.arm.infofract` *information-fraction based arm efficacy stop*

Description

allows stopping an arm for efficacy at a given look when the probability of the corresponding target parameter being greater or smaller (depending on the argument `'alternative'` of batss.glm) than `delta.eff` is greater than a function of the information fraction at that look.

Usage

```
eff.arm.infofract(posterior, b, n, N, p)
```

Arguments

`posterior`	the 'BATSS' ingredient '`posterior`' corresponding, in this context, to the (posterior) probability of the target parameter being greater or smaller (depending on the argument `'alternative'` of batss.glm) than '`delta.eff`'.
`b`	a tuning parameter (to be defined in `eff.arm.control`).

n	the 'BATSS' ingredient 'n' corresponding to the vector of number of recruited participants per arm including the control group.
N	the 'BATSS' ingredient ''N' corresponding to the maximum (planned) sample size.
p	a tuning parameter (to be defined in eff.arm.control).

Value

eff.arm.infofract returns a logical constant.

`eff.arm.simple` *Simple arm efficacy stop*

Description

allows stopping an arm for efficacy at a given look when the probability of the corresponding target parameter being greater or smaller (depending on the argument 'alternative' of batss.glm) than delta.eff is greater than a fixed value b.

Usage

```
eff.arm.simple(posterior, b)
```

Arguments

posterior	the 'BATSS' ingredient 'posterior' corresponding, in this context, to the (posterior) probability of the target parameter being greater or smaller (depending on the argument 'alternative' of batss.glm) than 'delta.eff'.
b	the cut-off value used to declare efficacy (to be defined in eff.arm.control).

Value

eff.arm.simple returns a logical constant.

`eff.trial.all` *trial efficacy stop*

Description

allows stopping the trial for efficacy if *all* target parameters reached efficacy at the look of interest or before.

Usage

```
eff.trial.all(eff.target)
```

Arguments

`eff.target`	the 'BATSS' ingredient '`eff.target`' corresponding to a logical vector of the same length as argument `which` (i.e., the number of target parameters) indicating if efficacy was reached for each target parameter at that stage or at a previous stage.

Value

eff.trial.all returns a logical constant.

`eff.trial.any` *trial efficacy stop*

Description

allows stopping the trial for efficacy if *at least one* target parameter reached efficacy at the look of interest.

Usage

```
eff.trial.any(eff.target)
```

Arguments

`eff.target`	the 'BATSS' ingredient '`eff.target`' corresponding to a logical vector of the same length as argument `which` (i.e., the number of target parameters) indicating if efficacy was reached for each target parameter at that stage or at a previous stage.

Value

eff.trial.any returns a logical constant.

`fut.arm.simple` *arm futility stop*

Description

allows stopping an arm for futility when the probability of the corresponding target parameter being greater or smaller (depending on the argument `'alternative'` of batss.glm) than '`delta.fut`' is smaller than a fixed value 'b'

Usage

```
fut.arm.simple(posterior, b)
```

Arguments

`posterior`	the 'BATSS' ingredient '`posterior`' corresponding, in this context, to the (posterior) probability of the target parameter being greater or smaller (depending on the argument '`alternative`' of batss.glm) than '`delta.fut`'.
`b`	the cut-off value used to declare futility (to be defined in `fut.arm.control`).

Value

fut.arm.simple returns a logical constant.

`fut.trial.all` *trial futility stop*

Description

allows stopping the trial for efficacy if *all* active treatment reached futility at the look of interest or before.

Usage

```
fut.trial.all(fut.target)
```

Arguments

`fut.target`	the 'BATSS' ingredient '`fut.target`' corresponding to a logical vector of the same length as argument `which` (i.e., the number of target parameters) indicating if futility was declared for each target parameter at that stage or at a previous stage.

Value

fut.trial.all returns a logical constant.

`plot.batss` *Plot function for 'BATSS' outputs*

Description

Plot for objects of class 'batss'

Usage

```
## S3 method for class 'batss'
plot(
  x,
  which = 1:4,
  ask = TRUE,
  hypothesis = "H1",
  title = TRUE,
  legend = TRUE,
  ess = TRUE,
  percentage = TRUE,
  beta = TRUE,
  col = c("#008B0040", "#8B3A3A40", "#8B897040", "#FF990075"),
  smooth = 1,
  ...
)
```

Arguments

x	An object of class 'batss' (i.e., output of the function batss.glm).
which	An integer vector indicating the list of desired plots. If a subset of the plots is required, specify a subset of the numbers 1:4. By default, all plots are provided, i.e., which=1:4. Plot 1 displays the boxplot of the total sample size as well as the boxplot of the sample sizes per group, Plot 2 displays a barplot of the probability of stopping at each look, plot 3 displays the violin plot of the sample size per group, and plot 4 displays the Monte Carlo trial target estimates as a function of the sample size.
ask	A logical. If TRUE, the user is prompted to hit the Enter key before each plot. See par(ask=.). The default is ask=TRUE.
hypothesis	A character string indicating which alternative hypothesis to use for analyses considering both "H0" and "H1", with options "H1" (default) and "H0".
title	Either a logical indicating if a title should be added or a string (of class character) indicating the title to be added. If title equals TRUE (default), the title 'Under 'H1' or 'Under 'H0' (depending on the argument hypothesis) is added to the outer margin of the plot. No outer margin space is added if title = FALSE.
legend	A logical (with default set to TRUE) if a legend should be added at the bottom of plots 3 and 4, or a list with names height, cex and pt respectively indicating i/ the fraction of the plot to be used for the legend as a numeric (default is .15), ii/ the character expansion factor relative to current par("cex") as a numeric (default to 1.25), and iii/ the expansion factor(s) for the points as a numeric (default to 2). The input legend = TRUE is equivalent to legend = c(height=.15, cex=1, pt=2).
ess	A logical (with default set to TRUE) indicating if the expected sample size should be displayed in plots 2, 3 and 4, or a list with names col, cex and bg respectively indicating i/ the colour of the label as a character (plots 2, 3, and 4), ii/

the text expension level as a numerical value (plots 2, 3 and 4) and iii/ the background colour as a character (plot 3). The input `ess = TRUE` is equivalent to `ess = list(col="blue", cex=1, bg="#FFD70070")`.

`percentage` A logical (with default set to `TRUE`) indicating if the probability of stopping at each look should be displayed in plots 2 (as a percentage), or a list with names `col`, and `cex` indicating i/ the colour of the label as a character, ii/ the text expension level as a numerical value. The input `percentage = TRUE` is equivalent to `percentage = list(col="violet", cex=1)`.

`beta` A logical (with default set to `TRUE`) indicating if the assumed target parameter value(s) should be displayed in plot 4, or a list with names `col`, `lwd`, and `lty` respectively indicating i/ the colour of the target parameter value horizontal line(s) as a character, ii/ the thickness of the line(s) as a numerical value and iii/ the type of line as an integer (see par for details). The input `beta = TRUE` is equivalent to `beta = list(col="blue", lwd=1.5, lty=1)`.

`col` A vector of length 4 specifying the colours to be used. Default to `c("#8B897040","#008B0040","#8B3A3`
where the last two digits of the hexadecimal strings specify the level of transparency. Refer to the Section 'colour specification' in par for details. If the length of `col` equals 1, the same colour is used for all cases. For plots 1 and 2, the 3rd colour of the vector `col` is used to display the barplot and boxplots.

`smooth` A numerical (>0) indictating the level of smoothing of the violin plots (for plot 3). Default to 1. When `smooth=NULL`, the smoothing value is optimised in the sm.density function.

`...` Additional arguments affecting the plot produced, like ylim and ylab.

Value

Generates graphical displays of results for objects of class 'batss'.

See Also

`batss.glm()`, the function generating S3 objects of class 'batss'.

`print.batss` *Print function for BATSS outputs*

Description

Print method function for objects of class 'batss' (i.e., output of the function batss.glm).

Usage

```
## S3 method for class 'batss'
print(x, ...)
```

Arguments

x	An object of class 'batss'.
...	Additional arguments affecting the print produced.

Value

Prints information for objects of class 'batss'.

See Also

batss.glm(), the function generating S3 objects of class 'batss'.

---

print.summary.batss *Print function for objects of class 'summary.batss'*

Description

Print function for objects of class 'summary.batss'

Usage

```
## S3 method for class 'summary.batss'
print(x, ...)
```

Arguments

x	An object of class 'summary.batss' (i.e., output of the function summary used on an output of the function batss.glm).
...	Additional arguments affecting the summary produced.

Value

Prints a summary for objects of class 'batss'.

See Also

batss.glm(), the function generating S3 objects of class 'batss'.

RAR.optimal *'Optimal' control allocation*

Description

technically not response adaptive but keeps allocation ratio to control at the square root of active intervention arms

Usage

```
RAR.optimal(active)
```

Arguments

active	the 'BATSS' ingredient 'active' corresponding to a logical vector of the same length and order as 'prob0' (i.e., number of arms initially included in the study including the reference group)) and indicating if each arm is active at the look of interest.

Value

RAR.optimal returns a vector of probabilities with length of active.

RAR.trippa *RAR of Trippa et al. (2012)*

Description

define the group allocation probabilities based on the response adaptive randomisation rule of Trippa et al. (2012)

Usage

```
RAR.trippa(posterior, n, N, ref, active, gamma, eta, nu)
```

Arguments

posterior	the 'BATSS' ingredient 'posterior' corresponding, in this context, to the (posterior) probability of the active target parameters being greater or smaller (depending on the argument 'alternative' of batss.glm) than 'delta.RAR'.
n	the 'BATSS' ingredient 'n' corresponding to the vector of number of recruited participants per arm including the control group at the look of interest.
N	the 'BATSS' ingredient ''N' corresponding to the maximum (planned) sample size.

ref	the 'BATSS' ingredient 'ref' corresponding to a logical vector of the same length and order as 'prob0' (i.e., number of arms initially included in the study including the reference group)) and indicating which group is the reference one.
active	the 'BATSS' ingredient 'active' corresponding to a logical vector of the same length and order as 'prob0' (i.e., number of arms initially included in the study including the reference group)) and indicating if each arm is active at the look of interest.
gamma	a scaling factor (to be defined in RAR.arm.control).
eta	a scaling factor (to be defined in RAR.arm.control).
nu	a scaling factor (to be defined in RAR.arm.control).

Value

RAR.trippa returns a vector of probabilities with length of active.

---

summary.batss *Summary function for 'BATSS' outputs*

Description

Summary method function for objects of class 'batss'.

Usage

```
## S3 method for class 'batss'
summary(object, extended = NULL, ...)
```

Arguments

object	An object of class 'batss' (i.e., output of the function batss.glm).
extended	A logical indicating if a standard (extended = FALSE, default) or extended output (extended = TRUE) should be returned. Default to NULL in which case the input of the argument extended chosen when generating object with batss.glm() is used.
...	Additional arguments affecting the summary produced.

Value

Object of class 'summary.batss'.

The function summary.batss returns an S3 list of class 'summary.batss' with available print functions. The list elements are

- beta - A data frame providing information related to the beta parameter vector, such as parameter names and values, for example.
- look - A data frame providing information related to looks, like sample size of a given interim (m) and cumulative sample size at a given interim (n), for example.

- par - A list providing different information, like the used seeds (seed) and the groups (group), for example.
- H1 - A list providing trial aggregated results under the alternative, like the probability of efficacy, futility, or both, per arm or globally (`object$H1$target`), the probability of stopping early for efficacy (`object$H1$efficacy`) and futility (`object$H1$futility`), the sample size expectation, standard deviation, and quantiles 0.1, 0.5 and 0.9, per group and overall (`object$H1$summary.sample.sizes`), the probabilities associated to each combination of efficacy and futility per group (scenario).
- H0 - A list providing trial aggregated results under the global null hypothesis (same structure as H1).
- call - The matched call.
- type - The type of 'BATSS' analysis (only 'glm' is currently available).

See Also

`batss.glm()`, the function generating S3 objects of class 'batss'.

`violin.density` *Violin plot density estimates*

Description

Extracts density estimates for a violin plot.

Usage

```
violin.density(data, h = NULL)
```

Arguments

`data`	data vector.
`h`	a scalar corresponding to the smoothing parameter.

Value

a list of coordinates and corresponding density values.

See Also

sm.density

Index

alloc.balanced, 2, *2*
alloc.balanced(), *3*
alloc.simple, *2*, 3, *3*
alloc.simple(), *2*

BATSS, 3
batss.combine, 4, *8*
batss.glm, *4*, 5, *8–17*
batss.glm(), *4*, *14, 15*, *17, 18*

call, *4*
character, *13*

eff.arm.infofract, *7*, 9, *10*
eff.arm.simple, *7*, 10, *10*
eff.trial.all, *7*, 10, *11*
eff.trial.any, *7*, 11, *11*

factor, *2, 3*, *6*
formula, *5*
fut.arm.simple, *7*, 11, *12*
fut.trial.all, *7*, 12, *12*

getOption, *8*
glm, *5*

identical, *4*
inla, *8*
inla.list.models, *6*

lm, *5*
logical, *4*, *7*, *11–14*, *16, 17*

model.matrix, *6*

par, *13, 14*
plot.batss, *8*, 12
print.batss, 14
print.summary.batss, 15

RAR.optimal, *7*, 16, *16*
RAR.trippa, *7*, 16, *17*
rexp, *6*
rnorm, *6*

set.seed, *6*
sm.density, *14*, *18*
summary, *15*
summary.batss, *8*, 17, *17*

violin.density, 18