Package 'BayesFactor'

July 21, 2025

Type Package

Title Computation of Bayes Factors for Common Designs

Version 0.9.12-4.7

Date 2024-01-23

Description A suite of functions for computing various Bayes factors for simple designs, including contingency tables, one- and two-sample designs, one-way designs, general ANOVA designs, and linear regression.

License GPL-2

VignetteBuilder knitr

Depends R (>= 3.2.0), coda, Matrix (>= 1.1-1)

Imports pbapply, mvtnorm, stringr, utils, graphics, MatrixModels, Rcpp (>= 0.11.2), methods, hypergeo

Suggests doMC, foreach, testthat, knitr, markdown, rmarkdown, arm, lme4, xtable, languageR

URL https://richarddmorey.github.io/BayesFactor/

BugReports https://github.com/richarddmorey/BayesFactor/issues

LazyLoad yes

LinkingTo Rcpp (>= 0.11.2), RcppEigen (>= 0.3.2.2.0)

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Author Richard D. Morey [aut, cre, cph],
Jeffrey N. Rouder [aut],
Tahira Jamil [ctb, cph],
Simon Urbanek [ctb, cph],
Karl Forner [ctb, cph],
Alexander Ly [ctb, cph]

Maintainer Richard D. Morey <richarddmorey@gmail.com>

Repository CRAN

Date/Publication 2024-01-24 15:02:50 UTC

Contents

BayesFactor-package *Functions to compute Bayes factor hypothesis tests for common research designs and hypotheses.*

Description

This package contains function to compute Bayes factors for a number of research designs and hypotheses, including t tests, ANOVA, and linear regression, correlations, proportions, and contingency tables.

Details

Package:	BayesFactor
Type:	Package
Version:	0.9.12-4.7
Date:	2024-01-23
License:	GPL 2.0
LazyLoad:	yes

The following methods are currently implemented, with more to follow:

general linear models (including linear mixed effects models): `generalTestBF`, `lmBF`

linear regression: `regressionBF`, `lmBF`, `linearReg.R2stat`;

linear correlation: `correlationBF`;

t tests: `ttestBF`, `ttest.tstat`;

meta-analytic t tests: `meta.ttestBF`

ANOVA: `anovaBF`, `lmBF`, `oneWayAOV.Fstat`;

contingency tables: `contingencyTableBF`;

single proportions: `proportionBF`;

linear correlations: `correlationBF`;

Other useful functions: `posterior`, for sampling from posterior distributions; `recompute`, for re-estimating a Bayes factor or posterior distribution; `compare`, to compare two model posteriors; and `plot.BFBayesFactor`, for plotting Bayes factor objects.

Author(s)

Richard D. Morey and Jeffrey N. Rouder (with contributions from Tahira Jamil)

Maintainer: Richard D. Morey <richarddmorey@gmail.com>

References

Liang, F. and Paulo, R. and Molina, G. and Clyde, M. A. and Berger, J. O. (2008). Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association, 103, pp. 410-423

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., and Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. Psychonomic Bulletin & Review, 16, 225-237

Rouder, J. N., Morey, R. D., Speckman, P. L., Province, J. M., (2012) Default Bayes Factors for ANOVA Designs. Journal of Mathematical Psychology. 56. p. 356-374.

Examples

```
## See specific functions for examples.
```

---

anovaBF *Function to compute Bayes factors for ANOVA designs*

Description

This function computes Bayes factors for all main-effects and interaction contrasts in an ANOVA design.

Usage

```
anovaBF(
  formula,
  data,
  whichRandom = NULL,
  whichModels = "withmain",
  iterations = 10000,
  progress = getOption("BFprogress", interactive()),
  rscaleFixed = "medium",
  rscaleRandom = "nuisance",
  rscaleEffects = NULL,
  multicore = FALSE,
  method = "auto",
  noSample = FALSE,
  callback = function(...) as.integer(0)
)
```

Arguments

formula	a formula containing all factors to include in the analysis (see Examples)
data	a data frame containing data for all factors in the formula
whichRandom	a character vector specifying which factors are random

whichModels	which set of models to compare; see Details
iterations	How many Monte Carlo simulations to generate, if relevant
progress	if TRUE, show progress with a text progress bar
rscaleFixed	prior scale for standardized, reduced fixed effects. A number of preset values can be given as strings; see Details.
rscaleRandom	prior scale for standardized random effects
rscaleEffects	A named vector of prior settings for individual factors, overriding rscaleFixed and rscaleRandom. Values are scales, names are factor names.
multicore	if TRUE use multiple cores through the doMC package. Unavailable on Windows.
method	approximation method, if needed. See nWayAOV for details.
noSample	if TRUE, do not sample, instead returning NA.
callback	callback function for third-party interfaces

Details

Models, priors, and methods of computation are provided in Rouder et al. (2012).

The ANOVA model for a vector of observations y is

$$y = \mu + X_1\theta_1 + \ldots + X_p\theta_p + \epsilon,$$

where $\theta_1, \ldots, \theta_p$ are vectors of main-effect and interaction effects, $X_1, \ldots, X_p$ are corresponding design matrices, and ϵ is a vector of zero-centered noise terms with variance σ^2. Zellner and Siow (1980) inspired g-priors are placed on effects, but with a separate g-prior parameter for each covariate:

$$\theta_1\ N(0, g_1\sigma^2), \ldots, \theta_p\ N(0, g_p\sigma^2).$$

A Jeffries prior is placed on μ and σ^2. Independent scaled inverse-chi-square priors with one degree of freedom are placed on $g_1, \ldots, g_p$. The square-root of the scale for g's corresponding to fixed and random effects is given by rscaleFixed and rscaleRandom, respectively.

When a factor is treated as random, there are as many main effect terms in the vector θ as levels. When a factor is treated as fixed, the sums-to-zero linear constraint is enforced by centering the corresponding design matrix, and there is one fewer main effect terms as levels. The Cornfield-Tukey model of interactions is assumed. Details are provided in Rouder et al. (2012)

Bayes factors are computed by integrating the likelihood with respect to the priors on parameters. The integration of all parameters except $g_1, \ldots, g_p$ may be expressed in closed-form; the integration of $g_1, \ldots, g_p$ is performed through Monte Carlo sampling, and iterations is the number of iterations used to estimate the Bayes factor.

anovaBF computes Bayes factors for either all submodels or select submodels missing a single main effect or covariate, depending on the argument whichModels. If no random factors are specified, the null model assumed by anovaBF is the grand-mean only model. If random factors are specified, the null model is the model with an additive model on all random factors, plus a grand mean. Thus, anovaBF does not currently test random factors. Testing random factors is possible with lmBF.

The argument whichModels controls which models are tested. Possible values are 'all', 'withmain', 'top', and 'bottom'. Setting whichModels to 'all' will test all models that can be created by including or not including a main effect or interaction. 'top' will test all models that can be created by

removing or leaving in a main effect or interaction term from the full model. 'bottom' creates models by adding single factors or interactions to the null model. 'withmain' will test all models, with the constraint that if an interaction is included, the corresponding main effects are also included.

For the `rscaleFixed` and `rscaleRandom` arguments, several named values are recognized: "medium", "wide", and "ultrawide", corresponding to r scale values of 1/2, $\sqrt{2}/2$, and 1, respectively. In addition, `rscaleRandom` can be set to the "nuisance", which sets $r = 1$ (and is thus equivalent to "ultrawide"). The "nuisance" setting is for medium-to-large-sized effects assumed to be in the data but typically not of interest, such as variance due to participants.

Value

An object of class `BFBayesFactor`, containing the computed model comparisons. Bayes factors can be extracted using extractBF(), as.vector() or as.data.frame().

Note

The function `anovaBF` will compute Bayes factors for all possible combinations of fixed factors and interactions, against the null hypothesis that *all* effects are 0. The total number of tests computed will be 2^{2^K-1} for K fixed factors. This number increases very quickly with the number of factors. For instance, for a five-way ANOVA, the total number of tests exceeds two billion. Even though each test takes a fraction of a second, the time taken for all tests could exceed your lifetime. An option is included to prevent this: `options('BFMaxModels')`, which defaults to 50,000, is the maximum number of models that 'anovaBF' will analyze at once. This can be increased by increasing the option value.

It is possible to reduce the number of models tested by only testing the most complex model and every restriction that can be formed by removing one factor or interaction using the `whichModels` argument. Setting this argument to 'top' reduces the number of tests to $2^K - 1$, which is more manageable. The Bayes factor for each restriction against the most complex model can be interpreted as a test of the removed factor/interaction. Setting `whichModels` to 'withmain' will not reduce the number of tests as much as 'top' but the results may be more interpretable, since an interaction is only allowed when all interacting effects (main or interaction) are also included in the model.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Gelman, A. (2005) Analysis of Variance—why it is more important than ever. Annals of Statistics, 33, pp. 1-53.

Rouder, J. N., Morey, R. D., Speckman, P. L., Province, J. M., (2012) Default Bayes Factors for ANOVA Designs. Journal of Mathematical Psychology. 56. p. 356-374.

Zellner, A. and Siow, A., (1980) Posterior Odds Ratios for Selected Regression Hypotheses. In Bayesian Statistics: Proceedings of the First Interanational Meeting held in Valencia (Spain). Bernardo, J. M., Lindley, D. V., and Smith A. F. M. (eds), pp. 585-603. University of Valencia.

See Also

lmBF, for testing specific models, and regressionBF for the function similar to anovaBF for linear regression models.

Examples

```
## Classical example, taken from t.test() example
## Student's sleep data
data(sleep)
plot(extra ~ group, data = sleep)

## traditional ANOVA gives a p value of 0.00283
summary(aov(extra ~ group + Error(ID/group), data = sleep))

## Gives a Bayes factor of about 11.6
## in favor of the alternative hypothesis
anovaBF(extra ~ group + ID, data = sleep, whichRandom = "ID",
    progress=FALSE)

## Demonstrate top-down testing
data(puzzles)
result = anovaBF(RT ~ shape*color + ID, data = puzzles, whichRandom = "ID",
    whichModels = 'top', progress=FALSE)
result

## In orthogonal designs, the top down Bayes factor can be
## interpreted as a test of the omitted effect
```

---

as.BFBayesFactor *Function to coerce objects to the BFBayesFactor class*

Description

This function coerces objects to the BFBayesFactor class

Usage

```
as.BFBayesFactor(object)
```

Arguments

object	an object of appropriate class (for now, BFBayesFactorTop)

Details

Function to coerce objects to the BFBayesFactor class

Currently, this function will only work with objects of class BFBayesFactorTop, which are output from the functions anovaBF and regressionBF when the whichModels argument is set to 'top'

Value

An object of class BFBayesFactor

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

See Also

regressionBF, anovaBF whose output is appropriate for use with this function when whichModels='top'

---

as.BFprobability *Function to coerce objects to the BFprobability class*

Description

This function coerces objects to the BFprobability class

Usage

```
as.BFprobability(object, normalize = NULL, lognormalize = NULL)
```

Arguments

object	an object of appropriate class (BFodds)
normalize	the sum of the probabilities for all models in the object (1 by default)
lognormalize	alternative to normalize; the logarithm of the normalization constant (0 by default)

Details

Function to coerce objects to the BFprobability class

Currently, this function will only work with objects of class BFOdds.

Value

An object of class BFprobability

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

`BFBayesFactor-class` *General S4 class for representing multiple Bayes factor model comparisons, all against the same model*

Description

The `BFBayesFactor` class is a general S4 class for representing models model comparison via Bayes factor.

Usage

```
## S4 method for signature 'numeric,BFBayesFactor'
e1 / e2

## S4 method for signature 'BFBayesFactor,BFBayesFactor'
e1 / e2

## S4 method for signature 'BFBayesFactor,index,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'BFBayesFactor'
t(x)

## S4 method for signature 'BFBayesFactor'
which.max(x)

## S4 method for signature 'BFBayesFactor'
which.min(x)

## S4 method for signature 'BFBayesFactor'
is.na(x)

## S4 method for signature 'BFBayesFactor,BFodds'
e1 * e2

## S4 method for signature 'BFBayesFactorTop,index,missing,missing'
x[i, j, ..., drop = TRUE]
```

Arguments

`e1`	Numerator of the ratio
`e2`	Denominator of the ratio
`x`	BFBayesFactor object
`i`	indices indicating elements to extract
`j`	unused for BFBayesFactor objects
`...`	further arguments passed to related methods
`drop`	unused

Details

BFBayesFactor objects can be inverted by taking the reciprocal and can be divided by one another, provided both objects have the same denominator. In addition, the t (transpose) method can be used to invert Bayes factor objects.

The BFBayesFactor class has the following slots defined:

numerator a list of models all inheriting BFmodel, each providing a single denominator

denominator a single BFmodel object serving as the denominator for all model comparisons

bayesFactor a data frame containing information about the comparison between each numerator and the denominator

data a data frame containing the data used for the comparison

version character string giving the version and revision number of the package that the model was created in

Examples

```
## Compute some Bayes factors to demonstrate division and indexing
data(puzzles)
bfs <- anovaBF(RT ~ shape*color + ID, data = puzzles, whichRandom = "ID", progress=FALSE)

## First and second models can be separated; they remain BFBayesFactor objects
b1 = bfs[1]
b2 = bfs[2]
b1

## We can invert them, or divide them to obtain new model comparisons
1/b1
b1 / b2

## Use transpose to create a BFBayesFactorList
t(bfs)
```

BFBayesFactorList-class

General S4 class for representing a collection of Bayes factor model comprisons, each against a different denominator

---

Description

The BFBayesFactorList class is a general S4 class for representing models model comparison via Bayes factor. See the examples for demonstrations of BFBayesFactorList methods.

Usage

```
## S4 method for signature 'BFBayesFactorList'
t(x)
```

```
## S4 method for signature 'numeric,BFBayesFactorList'
e1 / e2

## S4 method for signature 'BFBayesFactorList,index,index,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'BFBayesFactorList,index,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'BFBayesFactorList,missing,index,missing'
x[i, j, ..., drop = TRUE]
```

Arguments

x	a BFBayesFactorList object
e1	Numerator of the ratio
e2	Denominator of the ratio
i	indices specifying rows to extract
j	indices specifying columns to extract
...	further arguments passed to related methods
drop	unused

Details

BFBayesFactorList objects inherit from lists, and contain a single slot:

character string giving the version and revision number of the package that the model was created in

Each element of the list contains a single "BFBayesFactor" object. Each element of the list must have the same numerators, in the same order, as all the others. The list object is displayed as a matrix of Bayes factors.

Examples

version

```
## Compute some Bayes factors to demonstrate Bayes factor lists
data(puzzles)
bfs <- anovaBF(RT ~ shape*color + ID, data = puzzles, whichRandom = "ID", progress=FALSE)

## Create a matrix of Bayes factors
bfList <- bfs / bfs
bfList

## Use indexing to select parts of the 'matrix'
bfList[1,]
bfList[,1]

## We can use the t (transpose) function as well, to get back a BFBayesFactor
t(bfList[2,])
```

```
## Or transpose the whole matrix
t(bfList)
```

BFInfo *Prints the version information for the BayesFactor package*

Description

Prints the version, revision, and date information for the BayesFactor package

Usage

```
BFInfo(print = TRUE)
```

Arguments

print	if TRUE, print version information to the console

Details

This function prints the version and revision information for the BayesFactor package.

Value

BFInfo returns a character string containing the version and revision number of the package..

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

BFManual *Opens the HTML manual for the BayesFactor package*

Description

This function opens the HTML manual for the BayesFactor package in whatever browser is configured.

Usage

```
BFManual()
```

Details

This function opens the HTML manual for the BayesFactor package in whatever browser is configured.

Value

BFManual returns NULL invisibly.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

BFmodel-class *General S4 classes for representing models for comparison*

Description

The BFmodel is a general S4 class for representing models for comparison. The more classes BFlinearModel, BFindepSample, and BFoneSample inherit directly from BFmodel.

Details

These model classes all have the following slots defined:

Model type

identifier a list uniquely identifying the model from other models of the same type

prior list giving appropriate prior settings for the model

dataTypes a character vector whose names are possible columns in the data; elements specify the corresponding data type, currently one of c("fixed","random","continuous")

shortName a short, readable identifying string

longName a longer, readable identifying string

analysis object storing information about a previous analysis of this model

version character string giving the version and revision number of the package that the model was created in

---

BFodds-class *General S4 class for representing multiple odds model comparisons, all against the same model*

Description

The BFodds class is a general S4 class for representing models model comparison via prior or posterior odds.

Usage

```
## S4 method for signature 'numeric,BFodds'
e1 / e2

## S4 method for signature 'BFodds,BFodds'
e1 / e2

## S4 method for signature 'BFodds,BFBayesFactor'
e1 * e2

## S4 method for signature 'BFodds,index,missing,missing'
x[i, j, ..., drop = TRUE]
```

Arguments

e1	Numerator of the ratio
e2	Denominator of the ratio
x	BFodds object
i	indices indicating elements to extract
j	unused for BFodds objects
...	further arguments passed to related methods
drop	unused

Details

BFodds objects can be inverted by taking the reciprocal and can be divided by one another, provided both objects have the same denominator. In addition, the t (transpose) method can be used to invert odds objects.

The BFodds class has the following slots defined:

numerator a list of models all inheriting BFmodel, each providing a single numerator

denominator a single BFmodel object serving as the denominator for all model comparisons

logodds a data frame containing information about the (log) prior odds between each numerator and the denominator

bayesFactor a BFBayesFactor object (possibly) containing the evidence from the data.

version character string giving the version and revision number of the package that the model was created in

BFprobability-class *General S4 class for representing multiple model probability comparisons*

Description

The BFprobability class is a general S4 class for representing models model comparison via prior or posterior probabilities.

Usage

```
## S4 method for signature 'BFprobability,numeric'
e1 / e2

## S4 method for signature 'BFprobability,numeric'
e1 - e2

## S4 method for signature 'BFprobability,index,missing,missing'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'BFprobability,character'
filterBF(x, name, perl = FALSE, fixed = FALSE, ...)
```

Arguments

e1	BFprobability object
e2	new normalization constant
x	BFprobability object
i	indices indicating elements to extract
j	unused for BFprobability objects
...	further arguments passed to related methods
drop	unused
name	regular expression to search name
perl	logical. Should perl-compatible regexps be used? See ?grepl for details.
fixed	logical. If TRUE, pattern is a string to be matched as is. See ?grepl for details.

Details

The BFprobability class has the following slots defined:

A BFodds object containing the models from which to compute the probabilities

normalize the sum of the probabilities of all models (will often be 1.0)

version character string giving the version and revision number of the package that the model was created in

compare *Compare two models, with respect to some data*

Description

This method is used primarily in the backend, and will only rarely be called by the end user. But see the examples below for a demonstration.

Usage

```
compare(numerator, denominator, data, ...)
```

Arguments

numerator	first model
denominator	second model (if omitted, compare to predefined null)
data	data for the comparison
...	arguments passed to and from related methods

Value

The compare function will return a model comparison object, typically a Bayes factor

Examples

```
## Sample from the posteriors for two models
data(puzzles)

## Main effects model; result is a BFmcmc object, inheriting
## mcmc from the coda package
mod1 = lmBF(RT ~ shape + color + ID, data = puzzles, whichRandom = "ID",
   progress = FALSE, posterior = TRUE, iterations = 1000)

plot(mod1)

## Full model
mod2 = lmBF(RT ~ shape*color + ID, data = puzzles, whichRandom = "ID",
   progress = FALSE, posterior = TRUE, iterations = 1000)

## Each BFmcmc object contains the model used to generate it, so we
## can compare them (data is not needed, it is contained in the objects):

compare(mod1, mod2)
```

`contingencyTableBF` *Function for Bayesian analysis of one- and two-sample designs*

Description

This function computes Bayes factors for contingency tables.

Usage

```
contingencyTableBF(
  x,
  sampleType,
  fixedMargin = NULL,
  priorConcentration = 1,
  posterior = FALSE,
  callback = function(...) as.integer(0),
  ...
)
```

Arguments

`x`	an m by n matrix of counts (integers m,n > 1)
`sampleType`	the sampling plan (see details)
`fixedMargin`	for the independent multinomial sampling plan, which margin is fixed ("rows" or "cols")
`priorConcentration`	prior concentration parameter, set to 1 by default (see details)
`posterior`	if `TRUE`, return samples from the posterior instead of Bayes factor
`callback`	callback function for third-party interfaces
`...`	further arguments to be passed to or from methods.

Details

The Bayes factor provided by `contingencyTableBF` tests the independence assumption in contingency tables under various sampling plans, each of which is described below. See Gunel and Dickey (1974) for more details.

For `sampleType="poisson"`, the sampling plan is assumed to be one in which observations occur as a poisson process with an overall rate, and then assignment to particular factor levels occurs with fixed probability. Under the null hypothesis, the assignments to the two factors are independent. Importantly, the total N is not fixed.

For `sampleType="jointMulti"` (joint multinomial), the sampling plan is assumed to be one in which the total N is fixed, and observations are assigned to cells with fixed probability. Under the null hypothesis, the assignments to the two factors are independent.

For `sampleType="indepMulti"` (independent multinomial), the sampling plan is assumed to be one in which row or column totals are fixed, and each row or column is assumed to be multinomially

distributed. Under the null hypothesis, each row or column is assumed to have the same multinomial probabilities. The fixed margin must be given by the `fixedMargin` argument.

For `sampleType="hypergeom"` (hypergeometric), the sampling plan is assumed to be one in which both the row and column totals are fixed. Under the null hypothesis, the cell counts are assumed to be governed by the hypergeometric distribution.

For all models, the argument `priorConcentration` indexes the expected deviation from the null hypothesis under the alternative, and corresponds to Gunel and Dickey's (1974) "a" parameter.

Value

If `posterior` is `FALSE`, an object of class `BFBayesFactor` containing the computed model comparisons is returned.

If `posterior` is `TRUE`, an object of class `BFmcmc`, containing MCMC samples from the posterior is returned.

Note

Posterior sampling for the hypergeometric model under the alternative has not yet been implemented.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

Tahira Jamil (<tahjamil@gmail.com>)

References

Gunel, E. and Dickey, J., (1974) Bayes Factors for Independence in Contingency Tables. Biometrika, 61, 545-557

Examples

```
## Hraba and Grant (1970) doll race data
data(raceDolls)

## Compute Bayes factor for independent binomial design, with
## columns as the fixed margin
bf = contingencyTableBF(raceDolls, sampleType = "indepMulti", fixedMargin = "cols")
bf

## Posterior distribution of difference in probabilities, under alternative
chains = posterior(bf, iterations = 10000)
sameRaceGivenWhite = chains[,"pi[1,1]"] / chains[,"pi[*,1]"]
sameRaceGivenBlack = chains[,"pi[1,2]"] / chains[,"pi[*,2]"]
hist(sameRaceGivenWhite - sameRaceGivenBlack, xlab = "Probability increase",
  main = "Increase in probability of child picking\nsame race doll (white - black)",
  freq=FALSE, yaxt='n')
box()
```

`correlationBF` *Function for Bayesian analysis of correlations*

Description

Bayes factors or posterior samples for correlations.

Usage

```
correlationBF(
  y,
  x,
  rscale = "medium",
  nullInterval = NULL,
  posterior = FALSE,
  callback = function(...) as.integer(0),
  ...
)
```

Arguments

`y`	first continuous variable
`x`	second continuous variable
`rscale`	prior scale. A number of preset values can be given as strings; see Details.
`nullInterval`	optional vector of length 2 containing lower and upper bounds of an interval hypothesis to test, in correlation units
`posterior`	if `TRUE`, return samples from the posterior instead of Bayes factor
`callback`	callback function for third-party interfaces
`...`	further arguments to be passed to or from methods.

Details

The Bayes factor provided by `ttestBF` tests the null hypothesis that the true linear correlation ρ between two samples (y and x) of size n from normal populations is equal to 0. The Bayes factor is based on Jeffreys (1961) test for linear correlation. Noninformative priors are assumed for the population means and variances of the two population; a shifted, scaled beta(1/rscale,1/rscale) prior distribution is assumed for ρ (note that `rscale` is called κ by Ly et al. 2015; we call it `rscale` for consistency with other BayesFactor functions).

For the `rscale` argument, several named values are recognized: "medium.narrow", "medium", "wide", and "ultrawide". These correspond to r scale values of $1/\sqrt{(27)}$, $1/3$, $1/\sqrt{(3)}$ and 1, respectively.

The Bayes factor is computed via several different methods.

Value

If posterior is FALSE, an object of class BFBayesFactor containing the computed model comparisons is returned. If nullInterval is defined, then two Bayes factors will be computed: The Bayes factor for the interval against the null hypothesis that the probability is 0, and the corresponding Bayes factor for the complement of the interval.

If posterior is TRUE, an object of class BFmcmc, containing MCMC samples from the posterior is returned.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Ly, A., Verhagen, A. J. & Wagenmakers, E.-J. (2015). Harold Jeffreys's Default Bayes Factor Hypothesis Tests: Explanation, Extension, and Application in Psychology. Journal of Mathematical Psychology, Available online 28 August 2015, https://dx.doi.org/10.1016/j.jmp.2015.06.004.

Jeffreys, H. (1961). Theory of probability, 3rd edn. Oxford, UK: Oxford University Press.

See Also

cor.test

Examples

```
bf = correlationBF(y = iris$Sepal.Length, x = iris$Sepal.Width)
bf
## Sample from the corresponding posterior distribution
samples = correlationBF(y = iris$Sepal.Length, x = iris$Sepal.Width,
          posterior = TRUE, iterations = 10000)
plot(samples[,"rho"])
```

---

extractBF *Extract the Bayes factor from an object*

Description

Extract the Bayes factor from an object

Usage

```
extractBF(x, logbf = FALSE, onlybf = FALSE)

## S4 method for signature 'BFBayesFactor'
extractBF(x, logbf = FALSE, onlybf = FALSE)
```

Arguments

x	object from which to extract the Bayes factors
logbf	return the logarithm of the Bayes factors
onlybf	return a vector of only the Bayes factors

Value

Returns an object containing Bayes factors extracted from the object

Examples

```
## Sample from the posteriors for two models
data(puzzles)

bf = lmBF(RT ~ shape*color + ID, data = puzzles, whichRandom="ID", progress=FALSE)

extractBF(bf)
```

---

extractOdds *Extract the odds from an object*

Description

Extract the odds from an object

Usage

```
extractOdds(x, logodds = FALSE, onlyodds = FALSE)

## S4 method for signature 'BFodds'
extractOdds(x, logodds = FALSE, onlyodds = FALSE)
```

Arguments

x	object from which to extract
logodds	return the logarithm
onlyodds	return a vector of only the odds

Value

Returns an object containing odds extracted from the object

extractProbabilities *Extract the probabilities from an object*

Description

Extract the probabilities from an object

Usage

```
extractProbabilities(x, logprobs = FALSE, onlyprobs = FALSE)

## S4 method for signature 'BFprobability'
extractProbabilities(x, logprobs = FALSE, onlyprobs = FALSE)
```

Arguments

x	object from which to extract
logprobs	return the logarithm
onlyprobs	return a vector of only the probabilities

Value

Returns an object containing probabilities extracted from the object

filterBF *Filter the elements of an object according to some pre-specified criteria*

Description

Filter the elements of an object according to some pre-specified criteria

Usage

```
filterBF(x, name, perl = FALSE, fixed = FALSE, ...)
```

Arguments

x	object
name	regular expression to search name
perl	logical. Should perl-compatible regexps be used? See ?grepl for details.
fixed	logical. If TRUE, pattern is a string to be matched as is. See ?grepl for details.
...	arguments passed to and from related methods

Value

Returns a filtered object

generalTestBF *Function to compute Bayes factors for general designs*

Description

This function computes Bayes factors corresponding to restrictions on a full model.

Usage

```
generalTestBF(
  formula,
  data,
  whichRandom = NULL,
  whichModels = "withmain",
  neverExclude = NULL,
  iterations = 10000,
  progress = getOption("BFprogress", interactive()),
  rscaleFixed = "medium",
  rscaleRandom = "nuisance",
  rscaleCont = "medium",
  rscaleEffects = NULL,
  multicore = FALSE,
  method = "auto",
  noSample = FALSE,
  callback = function(...) as.integer(0)
)
```

Arguments

formula	a formula containing the full model for the analysis (see Examples)
data	a data frame containing data for all factors in the formula
whichRandom	a character vector specifying which factors are random
whichModels	which set of models to compare; see Details
neverExclude	a character vector containing a regular expression (see help for regex for details) that indicates which terms to always keep in the analysis
iterations	How many Monte Carlo simulations to generate, if relevant
progress	if TRUE, show progress with a text progress bar
rscaleFixed	prior scale for standardized, reduced fixed effects. A number of preset values can be given as strings; see Details.
rscaleRandom	prior scale for standardized random effects
rscaleCont	prior scale for standardized slopes
rscaleEffects	A named vector of prior settings for individual factors, overriding rscaleFixed and rscaleRandom. Values are scales, names are factor names.
multicore	if TRUE use multiple cores through the doMC package. Unavailable on Windows.

method	approximation method, if needed. See nWayAOV for details.
noSample	if TRUE, do not sample, instead returning NA.
callback	callback function for third-party interfaces

Details

See the help for anovaBF and anovaBF or details.

Models, priors, and methods of computation are provided in Rouder et al. (2012) and Liang et al (2008).

Value

An object of class BFBayesFactor, containing the computed model comparisons

Note

The function generalTestBF can compute Bayes factors for all restrictions of a full model against the null hypothesis that all effects are 0. The total number of tests computed – if all tests are requested – will be $2^K - 1$ for K factors or covariates. This number increases very quickly with the number of tested predictors. An option is included to prevent testing too many models: options('BFMaxModels'), which defaults to 50,000, is the maximum number of models that will be analyzed at once. This can be increased by increased using options.

It is possible to reduce the number of models tested by only testing the most complex model and every restriction that can be formed by removing one factor or interaction using the whichModels argument. See the help for anovaBF for details.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Rouder, J. N., Morey, R. D., Speckman, P. L., Province, J. M., (2012) Default Bayes Factors for ANOVA Designs. Journal of Mathematical Psychology. 56. p. 356-374.

Liang, F. and Paulo, R. and Molina, G. and Clyde, M. A. and Berger, J. O. (2008). Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association, 103, pp. 410-423

See Also

lmBF, for testing specific models, and regressionBF and anovaBF for other functions for testing multiple models simultaneously.

Examples

```
## Puzzles example: see ?puzzles and ?anovaBF
data(puzzles)
## neverExclude argument makes sure that participant factor ID
## is in all models
result = generalTestBF(RT ~ shape*color + ID, data = puzzles, whichRandom = "ID",
neverExclude="ID", progress=FALSE)
result
```

---

linearReg.R2stat *Use R^2 statistic to compute Bayes factor for regression designs*

Description

Using the classical R^2 test statistic for (linear) regression designs, this function computes the corresponding Bayes factor test.

Usage

```
linearReg.R2stat(N, p, R2, rscale = "medium", simple = FALSE)
```

Arguments

N	number of observations
p	number of predictors in model, excluding intercept
R2	proportion of variance accounted for by the predictors, excluding intercept
rscale	numeric prior scale
simple	if TRUE, return only the Bayes factor

Details

This function can be used to compute the Bayes factor corresponding to a multiple regression, using the classical R^2 (coefficient of determination) statistic. It can be used when you don't have access to the full data set for analysis by lmBF, but you do have the test statistic.

For details about the model, see the help for regressionBF, and the references therein.

The Bayes factor is computed via Gaussian quadrature.

Value

If simple is TRUE, returns the Bayes factor (against the intercept-only null). If FALSE, the function returns a vector of length 3 containing the computed log(e) Bayes factor, along with a proportional error estimate on the Bayes factor and the method used to compute it.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>) and Jeffrey N. Rouder (<rouderj@missouri.edu>)

References

Liang, F. and Paulo, R. and Molina, G. and Clyde, M. A. and Berger, J. O. (2008). Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association, 103, pp. 410-423

Rouder, J. N. and Morey, R. D. (in press, Multivariate Behavioral Research). Bayesian testing in regression.

See Also

integrate, lm; see lmBF for the intended interface to this function, using the full data set.

Examples

```
## Use attitude data set
data(attitude)
## Scatterplot
lm1 = lm(rating~complaints,data=attitude)
plot(attitude$complaints,attitude$rating)
abline(lm1)
## Traditional analysis
## p value is highly significant
summary(lm1)

## Bayes factor
## The Bayes factor is over 400,000;
## the data strongly favor hypothesis that
## the slope is not 0.
result = linearReg.R2stat(30,1,0.6813)
exp(result[['bf']])
```

---

lmBF *Function to compute Bayes factors for specific linear models*

Description

This function computes Bayes factors, or samples from the posterior, of specific linear models (either ANOVA or regression).

Usage

```
lmBF(
  formula,
  data,
  whichRandom = NULL,
  rscaleFixed = "medium",
  rscaleRandom = "nuisance",
  rscaleCont = "medium",
  rscaleEffects = NULL,
```

```
  posterior = FALSE,
  progress = getOption("BFprogress", interactive()),
  ...
)
```

Arguments

formula	a formula containing all factors to include in the analysis (see Examples)
data	a data frame containing data for all factors in the formula
whichRandom	a character vector specifying which factors are random
rscaleFixed	prior scale for standardized, reduced fixed effects. A number of preset values can be given as strings; see Details.
rscaleRandom	prior scale for standardized random effects
rscaleCont	prior scale for standardized slopes. A number of preset values can be given as strings; see Details.
rscaleEffects	A named vector of prior settings for individual factors, overriding rscaleFixed and rscaleRandom. Values are scales, names are factor names.
posterior	if TRUE, return samples from the posterior distribution instead of the Bayes factor
progress	if TRUE, show progress with a text progress bar
...	further arguments to be passed to or from methods.

Details

This function provides an interface for computing Bayes factors for specific linear models against the intercept-only null; other tests may be obtained by computing two models and dividing their Bayes factors. Specifics about the priors for regression models – and possible settings for rscaleCont – can be found in the help for regressionBF; likewise, details for ANOVA models – and settings for rscaleFixed and rscaleRandom – can be found in the help for anovaBF.

Currently, the function does not allow for general linear models, containing both continuous and categorical predcitors, but this support will be added in the future.

Value

If posterior is FALSE, an object of class BFBayesFactor, containing the computed model comparisons is returned. Otherwise, an object of class BFmcmc, containing MCMC samples from the posterior is returned.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

See Also

regressionBF and anovaBF for testing many regression or ANOVA models simultaneously.

Examples

```
## Puzzles data; see ?puzzles for details
data(puzzles)
## Bayes factor of full model against null
bfFull = lmBF(RT ~ shape + color + shape:color + ID, data = puzzles, whichRandom = "ID")

## Bayes factor of main effects only against null
bfMain = lmBF(RT ~ shape + color + ID, data = puzzles, whichRandom = "ID")

## Compare the main-effects only model to the full model
bfMain / bfFull

## sample from the posterior of the full model
samples = lmBF(RT ~ shape + color + shape:color + ID,
           data = puzzles, whichRandom = "ID", posterior = TRUE,
           iterations = 1000)

## Aother way to sample from the posterior of the full model
samples2 = posterior(bfFull, iterations = 1000)
```

---

logMeanExpLogs *Functions to compute the logarithm of the mean (and cumulative means) of vectors of logarithms*

Description

Given a vector of numeric values of real values represented in log form, `logMeanExpLogs` computes the logarithm of the mean of the (exponentiated) values. `logCumMeanExpLogs` computes the logarithm of the cumulative mean.

Usage

```
logMeanExpLogs(v)
```

Arguments

v	A vector of (log) values

Details

Given a vector of values of log values *v*, one could compute `log(mean(exp(v)))` in R. However, exponentiating and summing will cause a loss of precision, and possibly an overflow. These functions use the identity

$$\log(e^a + e^b) = a + \log(1 + e^{b-a})$$

and the method of computing $\log(1 + e^x)$ that avoids overflow (see the references). The code is written in C for very fast computations.

Value

`logMeanExpLogs` returns a single value, `logCumMeanExpLogs` returns a vector of values of the same length as v, and `logSummaryStats` returns a list of the log mean, log variance, and cumulative log means.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

For details of the approximation of $\log(1+e^x)$ used to prevent loss of precision, see https://www.codeproject.com/Articles/25294/Avoiding-Overflow-Underflow-and-Loss-of-Precision and https://www.johndcook.com/blog/standard_deviation/.

Examples

```
# Sample 100 values
y = log(rexp(100,1))

# These will give the same value,
# since e^y is "small"
logMeanExpLogs(y)
log(mean(exp(y)))

# We can make e^x overflow by multiplying
# e^y by e^1000
largeVals = y + 1000

# This will return 1000 + log(mean(exp(y)))
logMeanExpLogs(largeVals)

# This will overflow
log(mean(exp(largeVals)))
```

`meta.ttestBF` *Function for Bayesian analysis of one- and two-sample designs*

Description

This function computes mata-analytic Bayes factors, or samples from the posterior, for one- and two-sample designs where multiple t values have been observed.

Usage

```
meta.ttestBF(
  t,
  n1,
  n2 = NULL,
  nullInterval = NULL,
  rscale = "medium",
  posterior = FALSE,
  callback = function(...) as.integer(0),
  ...
)
```

Arguments

t	a vector of t statistics
n1	a vector of sample sizes for the first (or only) condition
n2	a vector of sample sizes. If NULL, a one-sample design is assumed
nullInterval	optional vector of length 2 containing lower and upper bounds of an interval hypothesis to test, in standardized units
rscale	prior scale. A number of preset values can be given as strings; see Details.
posterior	if TRUE, return samples from the posterior instead of Bayes factor
callback	callback function for third-party interfaces
...	further arguments to be passed to or from methods.

Details

The Bayes factor provided by meta.ttestBF tests the null hypothesis that the true effect size (or alternatively, the noncentrality parameters) underlying a set of t statistics is 0. Specifically, the Bayes factor compares two hypotheses: that the standardized effect size is 0, or that the standardized effect size is not 0. Note that there is assumed to be a single, common effect size δ underlying all t statistics. For one-sample tests, the standardized effect size is $(\mu - \mu_0)/\sigma$; for two sample tests, the standardized effect size is $(\mu_2 - \mu_1)/\sigma$.

A Cauchy prior is placed on the standardized effect size. The rscale argument controls the scale of the prior distribution, with rscale=1 yielding a standard Cauchy prior. See the help for ttestBF and the references below for more details.

The Bayes factor is computed via Gaussian quadrature. Posterior samples are drawn via independent-candidate Metropolis-Hastings.

Value

If posterior is FALSE, an object of class BFBayesFactor containing the computed model comparisons is returned. If nullInterval is defined, then two Bayes factors will be computed: The Bayes factor for the interval against the null hypothesis that the standardized effect is 0, and the corresponding Bayes factor for the compliment of the interval.

If posterior is TRUE, an object of class BFmcmc, containing MCMC samples from the posterior is returned.

Note

To obtain the same Bayes factors as Rouder and Morey (2011), change the prior scale to 1.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Morey, R. D. & Rouder, J. N. (2011). Bayes Factor Approaches for Testing Interval Null Hypotheses. Psychological Methods, 16, 406-419

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. Psychonomic Bulletin & Review, 16, 225-237

Rouder, J. N. & Morey, R. D. (2011). A Bayes Factor Meta-Analysis of Bem's ESP Claim. Psychonomic Bulletin & Review, 18, 682-689

See Also

ttestBF

Examples

```
## Bem's (2010) data (see Rouder & Morey, 2011)
t=c(-.15,2.39,2.42,2.43)
N=c(100,150,97,99)

## Using rscale=1 and one-sided test to be
## consistent with Rouder & Morey (2011)
bf = meta.ttestBF(t, N, rscale=1, nullInterval=c(0, Inf))
bf[1]

## plot posterior distribution of delta, assuming alternative
## turn off progress bar for example
samples = posterior(bf[1], iterations = 1000, progress = FALSE)
## Note that posterior() respects the nullInterval
plot(samples)
summary(samples)
```

---

model.matrix,BFBayesFactor-method

Design matrices for Bayes factor linear models analyses.

Description

This function returns the design matrix used for computation of the Bayes factor for the numerator of a BFBayesFactor object. There must not be more than one numerator in the BFBayesFactor object.

Usage

```
## S4 method for signature 'BFBayesFactor'
model.matrix(object, ...)

## S4 method for signature 'BFBayesFactorTop'
model.matrix(object, ...)
```

Arguments

object	a BayesFactor object with a single numerator
...	arguments passed to and from related methods

Value

Returns the design matrix for the corresponding model. The 'gMap' attribute of the returned matrix contains the mapping from columns of the design matrix to g parameters

References

Rouder, J. N., Morey, R. D., Speckman, P. L., Province, J. M., (2012) Default Bayes Factors for ANOVA Designs. Journal of Mathematical Psychology. 56. p. 356-374.

Examples

```
## Gets the design matrix for a simple analysis
data(sleep)

bf = anovaBF(extra ~ group + ID, data = sleep, whichRandom="ID", progress=FALSE)
X = model.matrix(bf)

## Show dimensions of X (should be 20 by 12)
dim(X)
```

---

newPriorOdds *Create prior odds from a Bayes factor object*

Description

Create a prior odds object from a Bayes factor object

Usage

```
newPriorOdds(bf, type = "equal")
```

Arguments

bf	A BFBayesFactor object, eg, from an analysis
type	The type of prior odds to create (by default "equal"; see details)

Details

This function takes a Bayes factor object and, using its structure and specified type of prior odds, will create a prior odds object.

For now, the only type is "equal", which assigns equal prior odds to all models.

Value

A (prior) BFodds object, which can then be multiplied by the BFBayesFactor object to obtain posterior odds.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

`nWayAOV` *Use ANOVA design matrix to compute Bayes factors or sample posterior*

Description

Computes a single Bayes factor, or samples from the posterior, for an ANOVA model defined by a design matrix

Usage

```
nWayAOV(
  y,
  X,
  gMap,
  rscale,
  iterations = 10000,
  progress = getOption("BFprogress", interactive()),
  callback = function(...) as.integer(0),
  gibbs = NULL,
  posterior = FALSE,
  ignoreCols = NULL,
  thin = 1,
  method = "auto",
  continuous = FALSE,
  noSample = FALSE
)
```

Arguments

y	vector of observations
X	design matrix whose number of rows match length(y).
gMap	vector grouping the columns of X (see Details).
rscale	a vector of prior scale(s) of appropriate length (see Details).
iterations	Number of Monte Carlo samples used to estimate Bayes factor or posterior
progress	if TRUE, show progress with a text progress bar
callback	callback function for third-party interfaces
gibbs	will be deprecated. See posterior
posterior	if TRUE, return samples from the posterior using Gibbs sampling, instead of the Bayes factor
ignoreCols	if NULL and posterior=TRUE, all parameter estimates are returned in the MCMC object. If not NULL, a vector of length P-1 (where P is number of columns in the design matrix) giving which effect estimates to ignore in output
thin	MCMC chain to every thin iterations. Default of 1 means no thinning. Only used if posterior=TRUE
method	the integration method (only valid if posterior=FALSE); one of "simple", "importance", "laplace", or "auto"
continuous	either FALSE if no continuous covariates are included, or a logical vector of length equal to number of columns of X indicating which columns of the design matrix represent continuous covariates
noSample	if TRUE, do not sample, instead returning NA. This is intended to be used with functions generating and testing many models at one time, such as anovaBF

Details

This function is not meant to be called by end-users, although technically-minded users can call this function for flexibility beyond what the other functions in this package provide. See lmBF for a user-friendly front-end to this function. Details about the priors can be found in the help for anovaBF and the references therein.

Argument gMap provides a way of grouping columns of the design matrix as a factor; the effects in each group will share a common g parameter. gMap should be a vector of the same length as the number of nonconstant rows in X. It will contain all integers from 0 to $N_g - 1$, where N_g is the total number of g parameters. Each element of gMap specifies the group to which that column belongs.

If all columns belonging to a group are adjacent, struc can instead be used to compactly represent the groupings. struc is a vector of length N_g. Each element specifies the number columns in the group.

The vector rscale should be of length N_g, and contain the prior scales of the standardized effects. See Rouder et al. (2012) for more details and the help for anovaBF for some typical values.

The method used to estimate the Bayes factor depends on the method argument. "simple" is most accurate for small to moderate sample sizes, and uses the Monte Carlo sampling method described in Rouder et al. (2012). "importance" uses an importance sampling algorithm with an importance distribution that is multivariate normal on log(g). "laplace" does not sample, but uses a Laplace

approximation to the integral. It is expected to be more accurate for large sample sizes, where MC sampling is slow. If `method="auto"`, then an initial run with both samplers is done, and the sampling method that yields the least-variable samples is chosen. The number of initial test iterations is determined by `options(BFpretestIterations)`.

If posterior samples are requested, the posterior is sampled with a Gibbs sampler.

Value

If `posterior` is `FALSE`, a vector of length 2 containing the computed log(e) Bayes factor (against the intercept-only null), along with a proportional error estimate on the Bayes factor. Otherwise, an object of class `mcmc`, containing MCMC samples from the posterior is returned.

Note

Argument `struc` has been deprecated. Use `gMap`, which is the `inverse.rle` of `struc`, minus 1.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>), Jeffery N. Rouder (<rouderj@missouri.edu>)

References

Rouder, J. N., Morey, R. D., Speckman, P. L., Province, J. M., (2012) Default Bayes Factors for ANOVA Designs. Journal of Mathematical Psychology. 56. p. 356-374.

See Also

See `lmBF` for the user-friendly front end to this function; see `regressionBF` and `anovaBF` for testing many regression or ANOVA models simultaneously.

Examples

```
## Classical example, taken from t.test() example
## Student's sleep data
data(sleep)
plot(extra ~ group, data = sleep)

## traditional ANOVA gives a p value of 0.00283
summary(aov(extra ~ group + Error(ID/group), data = sleep))

## Build design matrix
group.column <- rep(1/c(-sqrt(2),sqrt(2)),each=10)
subject.matrix <- model.matrix(~sleep$ID - 1,data=sleep$ID)
## Note that we include no constant column
X <- cbind(group.column, subject.matrix)

## (log) Bayes factor of full model against grand-mean only model
bf.full <- nWayAOV(y = sleep$extra, X = X, gMap = c(0,rep(1,10)), rscale=c(.5,1))
exp(bf.full[['bf']])

## Compare with lmBF result (should be about the same, give or take 1%)
```

```
bf.full2 <- lmBF(extra ~ group + ID, data = sleep, whichRandom = "ID")
bf.full2
```

`oneWayAOV.Fstat` *Use F statistic to compute Bayes factor for balanced one-way designs*

Description

Using the classical F test statistic for a balanced one-way design, this function computes the corresponding Bayes factor test.

Usage

```
oneWayAOV.Fstat(F, N, J, rscale = "medium", simple = FALSE)
```

Arguments

`F`	F statistic from classical ANOVA
`N`	number of observations per cell or group
`J`	number of cells or groups
`rscale`	numeric prior scale
`simple`	if `TRUE`, return only the Bayes factor

Details

For F statistics computed from balanced one-way designs, this function can be used to compute the Bayes factor testing the model that all group means are not equal to the grand mean, versus the null model that all group means are equal. It can be used when you don't have access to the full data set for analysis by `lmBF`, but you do have the test statistic.

For details about the model, see the help for `anovaBF`, and the references therein.

The Bayes factor is computed via Gaussian quadrature.

Value

If `simple` is `TRUE`, returns the Bayes factor (against the intercept-only null). If `FALSE`, the function returns a vector of length 3 containing the computed log(e) Bayes factor, along with a proportional error estimate on the Bayes factor and the method used to compute it.

Note

`oneWayAOV.Fstat` should only be used with F values obtained from balanced designs.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Morey, R. D., Rouder, J. N., Pratte, M. S., and Speckman, P. L. (2011). Using MCMC chain outputs to efficiently estimate Bayes factors. Journal of Mathematical Psychology, 55, 368-378

See Also

integrate, aov; see lmBF for the intended interface to this function, using the full data set.

Examples

```
## Example data "InsectSprays" - see ?InsectSprays
require(stats); require(graphics)
boxplot(count ~ spray, data = InsectSprays, xlab = "Type of spray",
        ylab = "Insect count", main = "InsectSprays data", varwidth = TRUE,
        col = "lightgray")

## Classical analysis (with transformation)
classical <- aov(sqrt(count) ~ spray, data = InsectSprays)
plot(classical)
summary(classical)

## Bayes factor (a very large number)
Fvalue <- anova(classical)$"F value"[1]
result <- oneWayAOV.Fstat(Fvalue, N=12, J=6)
exp(result[['bf']])
```

---

options-BayesFactor *options() for package BayesFactor*

Description

Options that can be set for the BayesFactor package

Details

The BayesFactor package has numerous options that can be set to globally change the behavior of the functions in the package. These options can be changed using options().

`BFMaxModels` Integer; maximum number of models to analyze in anovaBF or regressionBF

`BFprogress` If `TRUE`, progress bars are on by default; if `FALSE`, they are disabled by default.

`BFpretestIterations` Integer; if sampling is needed to compute the Bayes factor, the package attempts to choose the most efficient sampler. This option controls the number of initial test iterations.

`BFapproxOptimizer` `"nlm"` or `"optim"`; changes the optimization function used for the importance sampler. If one fails, try the other.

`BFapproxLimits` Vector of length two containing the lower and upper limits on on `log(g)` before the the posterior returns `-Inf`. This only affects the initial optimization step for the importance sampler.

`BFfactorsMax` Maximum number of factors to try to do enumeration with in generalTestBF.

`BFcheckProbabilityList` Check for duplicate models when creating BFprobability objects?

See Also

`options`

`plot.BFBayesFactor` *Plot a Bayes factor object*

Description

Plot a Bayes factor object

Usage

```
## S3 method for class 'BFBayesFactor'
plot(
  x,
  include1 = TRUE,
  addDenom = FALSE,
  sortbf = TRUE,
  logbase = c("log10", "log2", "ln"),
  marginExpand = 0.4,
  cols = c("wheat", "lightslateblue"),
  main = paste("vs.", x@denominator@longName),
  pars = NULL,
  ...
)
```

Arguments

`x`	a BFBayesFactor object
`include1`	if `TRUE`, ensure that Bayes factor = 1 is on the plot
`addDenom`	if `TRUE`, add the denominator model into the group
`sortbf`	sort the Bayes factors before plotting them? Defaults to `TRUE`
`logbase`	the base of the log Bayes factors in the plot
`marginExpand`	an expansion factor for the left margin, in case more space is needed for model names
`cols`	a vector of length two of valid color names or numbers
`main`	a character vector for the plot title
`pars`	a list of par() settings
`...`	additional arguments to pass to barplot()

Details

This function creates a barplot of the (log) Bayes factors in a Bayes factor object. Error bars are added (though in many cases they may be too small to see) in red to show the error in estimation of the Bayes factor. If a red question mark appears next to a bar, then that Bayes factor has no error estimate available.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

Examples

```
data(puzzles)

bfs = anovaBF(RT ~ shape*color + ID, data = puzzles, whichRandom="ID", progress=FALSE)
plot(bfs)
```

---

plot.BFBayesFactorTop *Plot a Bayes factor top-down object*

Description

Plot a Bayes factor top-down object

Usage

```
## S3 method for class 'BFBayesFactorTop'
plot(
  x,
  include1 = TRUE,
  addDenom = FALSE,
  sortbf = FALSE,
  logbase = c("log10", "log2", "ln"),
  marginExpand = 0.4,
  pars = NULL,
  ...
)
```

Arguments

x	a BFBayesFactorTop object
include1	if TRUE, ensure that Bayes factor = 1 is on the plot
addDenom	if TRUE, add the denominator model into the group
sortbf	sort the Bayes factors before plotting them? Defaults to TRUE
logbase	the base of the log Bayes factors in the plot

marginExpand	an expansion factor for the left margin, in case more space is needed for model names
pars	a list of par() settings
...	additional arguments to pass to barplot()

Details

This function creates a barplot of the (log) Bayes factors in a Bayes factor object. Error bars are added (though in many cases they may be too small to see) in red to show the error in estimation of the Bayes factor. If a red question mark appears next to a bar, then that Bayes factor has no error estimate available.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

Examples

```
data(puzzles)

bfs = anovaBF(RT ~ shape*color + ID, data = puzzles, whichRandom="ID",
        whichModels='top', progress=FALSE)
plot(bfs)
```

---

posterior *Sample from the posterior distribution of one of several models.*

Description

This function samples from the posterior distribution of a BFmodel, which can be obtained from a BFBayesFactor object. If there is more than one numerator in the BFBayesFactor object, the index argument can be passed to select one numerator.

Usage

```
posterior(model, index, data, iterations, ...)

## S4 method for signature 'BFmodel,missing,data.frame,missing'
posterior(model, index, data, iterations, ...)

## S4 method for signature 'BFBayesFactor,missing,missing,missing'
posterior(model, index, data, iterations, ...)

## S4 method for signature 'BFBayesFactor,numeric,missing,numeric'
posterior(model, index, data, iterations, ...)

## S4 method for signature 'BFBayesFactor,missing,missing,numeric'
```

```
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFlinearModel,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFindepSample,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFcontingencyTable,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFoneSample,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFmetat,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFproportion,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)

## S4 method for signature 'BFcorrelation,missing,data.frame,numeric'
posterior(model, index = NULL, data, iterations, ...)
```

Arguments

`model`	or set of models from which to sample
`index`	the index within the set of models giving the desired model
`data`	the data to be conditioned on
`iterations`	the number of iterations to sample
`...`	arguments passed to and from related methods

Details

The data argument is used internally, and will y not be needed by end-users.

Note that if there are fixed effects in the model, the reduced parameterzation used internally (see help for anovaBF) is unreduced. For a factor with two levels, the chain will contain two effect estimates that sum to 0.

Two useful arguments that can be passed to related methods are `thin` and `columnFilter`, currently implemented for methods using `nWayAOV` (models with more than one categorical covariate, or a mix of categorical and continuous covariates). `thin`, an integer, will keep only every `thin` iterations. The default is `thin=1`, which keeps all iterations. Argument `columnFilter` is either `NULL` (for no filtering) or a character vector of extended regular expressions (see regex help for details). Any column from an effect that matches one of the filters will not be saved.

Value

Returns an object containing samples from the posterior distribution of the specified model

Examples

```
## Sample from the posteriors for two models
data(sleep)

bf = lmBF(extra ~ group + ID, data = sleep, whichRandom="ID", progress=FALSE)

## sample from the posterior of the numerator model
## data argument not needed - it is included in the Bayes factor object
chains = posterior(bf, iterations = 1000, progress = FALSE)

plot(chains)

## demonstrate column filtering by filtering out participant effects
data(puzzles)
bf = lmBF(RT ~ shape + color + shape:color + ID, data=puzzles)
chains = posterior(bf, iterations = 1000, progress = FALSE, columnFilter="^ID$")
colnames(chains) # Contains no participant effects
```

---

priorLogodds<- *Set prior log odds in an object*

Description

Set prior log odds in an object

Usage

```
priorLogodds(object) <- value

## S4 replacement method for signature 'BFodds,numeric'
priorLogodds(object) <- value
```

Arguments

object	object in which to set log odds
value	log odds

---

priorOdds<- *Set prior odds in an object*

Description

Set prior odds in an object

Usage

```
priorOdds(object) <- value

## S4 replacement method for signature 'BFodds,numeric'
priorOdds(object) <- value
```

Arguments

object	object in which to set odds
value	odds

---

proportionBF *Function for Bayesian analysis of proportions*

Description

Bayes factors or posterior samples for binomial, geometric, or neg. binomial data.

Usage

```
proportionBF(
  y,
  N,
  p,
  rscale = "medium",
  nullInterval = NULL,
  posterior = FALSE,
  callback = function(...) as.integer(0),
  ...
)
```

Arguments

y	a vector of successes
N	a vector of total number of observations
p	the null value for the probability of a success to be tested against
rscale	prior scale. A number of preset values can be given as strings; see Details.
nullInterval	optional vector of length 2 containing lower and upper bounds of an interval hypothesis to test, in probability units
posterior	if TRUE, return samples from the posterior instead of Bayes factor
callback	callback function for third-party interfaces
...	further arguments to be passed to or from methods.

Details

Given count data modeled as a binomial, geometric, or negative binomial random variable, the Bayes factor provided by `proportionBF` tests the null hypothesis that the probability of a success is p_0 (argument `p`). Specifically, the Bayes factor compares two hypotheses: that the probability is p_0, or probability is not p_0. Currently, the default alternative is that

$$\lambda \; logistic(\lambda_0, r)$$

where $\lambda_0 = logit(p_0)$ and $\lambda = logit(p)$. r serves as a prior scale parameter.

For the `rscale` argument, several named values are recognized: "medium", "wide", and "ultrawide". These correspond to r scale values of $1/2$, $\sqrt{2}/2$, and 1, respectively.

The Bayes factor is computed via Gaussian quadrature, and posterior samples are drawn via independence Metropolis-Hastings.

Value

If `posterior` is `FALSE`, an object of class `BFBayesFactor` containing the computed model comparisons is returned. If `nullInterval` is defined, then two Bayes factors will be computed: The Bayes factor for the interval against the null hypothesis that the probability is p_0, and the corresponding Bayes factor for the compliment of the interval.

If `posterior` is `TRUE`, an object of class `BFmcmc`, containing MCMC samples from the posterior is returned.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

See Also

`prop.test`

Examples

```
bf = proportionBF(y = 15, N = 25, p = .5)
bf
## Sample from the corresponding posterior distribution
samples =proportionBF(y = 15, N = 25, p = .5, posterior = TRUE, iterations = 10000)
plot(samples[,"p"])
```

---

puzzles *Puzzle completion times from Hays (1994)*

Description

Puzzle completion time example data from Hays (1994).

Format

A data frame with 48 observations on 3 variables.

RT Puzzle completion time, in minutes

ID the subject identifier

shape shape of the puzzle (round or square)

color color content of the puzzle (monochromatic or color)

Details

Hays (1994; section 13.21, table 13.21.2, p. 570) describes a experiment wherein 12 participants complete four puzzles each. Puzzles could be either square or round, and either monochromatic or in color. Each participant completed every combination of the two factors.

Source

Hays, W. L. (1994), Statistics (5th edition), Harcourt Brace, Fort Worth, Texas

Examples

```
data(puzzles)

## classical ANOVA
## Both color and shape are significant, interaction is not
classical <- aov(RT ~ shape*color + Error(ID/(shape*color)), data=puzzles)
summary(classical)

## Bayes Factor
## Best model is main effects model, no interaction
anovaBF(RT ~ shape*color + ID, data = puzzles, whichRandom = "ID", progress=FALSE)
```

---

raceDolls *Hraba and Grant (1970) children's doll preference data*

Description

Hraba and Grant (1970) describe a replication of Clark and Clark (1947) in which black and white children from Lincoln, Nebraska were shown dolls that were either black or white. They were then asked a series of questions, including "Give me the doll that is a nice doll." This data set contains the frequency of children giving the same-race or different race doll in response to this question.

Format

A matrix with 2 rows and 2 columns. Rows give doll preference; colums give the race of the child.

Source

Hraba, J. and Grant, G. (1970). Black is Beautiful: A reexamination of racial preference and identification. Journal of Personality and Social Psychology, 16, 398-402.

Examples

```
data(raceDolls)

## chi-square test
## Barely significant with continuity correction
chisq.test(raceDolls)

## Bayes factor test (assuming independent binomial sampling plan)
## Very little evidence for the alternative of lack of independence
bf = contingencyTableBF(raceDolls, sampleType = "indepMulti", fixedMargin = "cols")
bf
```

`recompute` *Recompute a Bayes factor computation or MCMC object.*

Description

Take an object and redo the computation (useful for sampling). In cases where sampling is used to compute the Bayes factor, the estimate of the precision of new samples will be added to the estimate precision of the old sample will be added to produce a new estimate of the precision.

Usage

```
recompute(
  x,
  progress = getOption("BFprogress", interactive()),
  multicore = FALSE,
  callback = function(...) as.integer(0),
  ...
)

## S4 method for signature 'BFBayesFactor'
recompute(
  x,
  progress = getOption("BFprogress", interactive()),
  multicore = FALSE,
  callback = function(...) as.integer(0),
  ...
)

## S4 method for signature 'BFBayesFactorTop'
recompute(
```

```
  x,
  progress = getOption("BFprogress", interactive()),
  multicore = FALSE,
  callback = function(...) as.integer(0),
  ...
)

## S4 method for signature 'BFmcmc'
recompute(
  x,
  progress = getOption("BFprogress", interactive()),
  multicore = FALSE,
  callback = function(...) as.integer(0),
  ...
)

## S4 method for signature 'BFodds'
recompute(
  x,
  progress = getOption("BFprogress", interactive()),
  multicore = FALSE,
  callback = function(...) as.integer(0),
  ...
)
```

Arguments

x	object to recompute
progress	report progress of the computation?
multicore	Use multicore, if available
callback	callback function for third-party interfaces
...	arguments passed to and from related methods

Value

Returns an object of the same type, after repeating the sampling (perhaps with more iterations)

Examples

```
## Sample from the posteriors for two models
data(puzzles)

## Main effects model; result is a BFmcmc object, inheriting
## mcmc from the coda package
bf = lmBF(RT ~ shape + color + ID, data = puzzles, whichRandom = "ID",
   progress = FALSE)

## recompute Bayes factor object
recompute(bf, iterations = 1000, progress = FALSE)
```

```
## Sample from posterior distribution of model above, and recompute:
chains = posterior(bf, iterations = 1000, progress = FALSE)
newChains = recompute(chains, iterations = 1000, progress=FALSE)
```

`regressionBF` *Function to compute Bayes factors for regression designs*

Description

This function simultaneously computes Bayes factors for groups of models in regression designs

Usage

```
regressionBF(
  formula,
  data,
  whichModels = "all",
  progress = getOption("BFprogress", interactive()),
  rscaleCont = "medium",
  callback = function(...) as.integer(0),
  noSample = FALSE
)
```

Arguments

`formula`	a formula containing all covariates to include in the analysis (see Examples)
`data`	a data frame containing data for all factors in the formula
`whichModels`	which set of models to compare; see Details
`progress`	if `TRUE`, show progress with a text progress bar
`rscaleCont`	prior scale on all standardized slopes
`callback`	callback function for third-party interfaces
`noSample`	if `TRUE`, do not sample, instead returning NA.

Details

`regressionBF` computes Bayes factors to test the hypothesis that slopes are 0 against the alternative that all slopes are nonzero.

The vector of observations y is assumed to be distributed as

$$y \; Normal(\alpha 1 + X\beta, \sigma^2 I).$$

The joint prior on α, σ^2 is proportional to $1/\sigma^2$, the prior on β is

$$\beta \; Normal(0, Ng\sigma^2(X'X)^{-1}).$$

where $g\ InverseGamma(1/2, r/2)$. See Liang et al. (2008) section 3 for details.

Possible values for `whichModels` are 'all', 'top', and 'bottom', where 'all' computes Bayes factors for all models, 'top' computes the Bayes factors for models that have one covariate missing from the full model, and 'bottom' computes the Bayes factors for all models containing a single covariate. Caution should be used when interpreting the results; when the results of 'top' testing is interpreted as a test of each covariate, the test is conditional on all other covariates being in the model (and likewise 'bottom' testing is conditional on no other covariates being in the model).

An option is included to prevent analyzing too many models at once: `options('BFMaxModels')`, which defaults to 50,000, is the maximum number of models that 'regressionBF' will analyze at once. This can be increased by increasing the option value.

For the `rscaleCont` argument, several named values are recongized: "medium", "wide", and "ultra-wide", which correspond r scales of $\sqrt{2}/4$, 1/2, and $\sqrt{2}/2$, respectively. These values were chosen to yield consistent Bayes factors with `anovaBF`.

Value

An object of class `BFBayesFactor`, containing the computed model comparisons

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Liang, F. and Paulo, R. and Molina, G. and Clyde, M. A. and Berger, J. O. (2008). Mixtures of g-priors for Bayesian Variable Selection. Journal of the American Statistical Association, 103, pp. 410-423

Rouder, J. N. and Morey, R. D. (in press). Bayesian testing in regression. Multivariate Behavioral Research.

Zellner, A. and Siow, A., (1980) Posterior Odds Ratios for Selected Regression Hypotheses. In Bayesian Statistics: Proceedings of the First Interanational Meeting held in Valencia (Spain). Bernardo, J. M., Lindley, D. V., and Smith A. F. M. (eds), pp. 585-603. University of Valencia.

See Also

`lmBF`, for testing specific models, and `anovaBF` for the function similar to `regressionBF` for ANOVA models.

Examples

```
## See help(attitude) for details about the data set
data(attitude)

## Classical regression
summary(fm1 <- lm(rating ~ ., data = attitude))

## Compute Bayes factors for all regression models
output = regressionBF(rating ~ ., data = attitude, progress=FALSE)
head(output)
```

```
## Best model is 'complaints' only

## Compute all Bayes factors against the full model, and
## look again at best models
head(output / output[63])
```

`ttest.tstat` *Use t statistic to compute Bayes factor for one- and two- sample designs*

Description

Using the classical t test statistic for a one- or two-sample design, this function computes the corresponding Bayes factor test.

Usage

```
ttest.tstat(
  t,
  n1,
  n2 = 0,
  nullInterval = NULL,
  rscale = "medium",
  complement = FALSE,
  simple = FALSE
)
```

Arguments

`t`	classical t statistic
`n1`	size of first group (or only group, for one-sample tests)
`n2`	size of second group, for independent-groups tests
`nullInterval`	optional vector of length 2 containing lower and upper bounds of an interval hypothesis to test, in standardized units
`rscale`	numeric prior scale
`complement`	if `TRUE`, compute the Bayes factor against the complement of the interval
`simple`	if `TRUE`, return only the Bayes factor

Details

This function can be used to compute the Bayes factor corresponding to a one-sample, a paired-sample, or an independent-groups t test, using the classical t statistic. It can be used when you don't have access to the full data set for analysis by `ttestBF`, but you do have the test statistic.

For details about the model, see the help for `ttestBF`, and the references therein.

The Bayes factor is computed via Gaussian quadrature.

Value

If simple is TRUE, returns the Bayes factor (against the null). If FALSE, the function returns a vector of length 3 containing the computed log(e) Bayes factor, along with a proportional error estimate on the Bayes factor and the method used to compute it.

Note

In version 0.9.9, the behaviour of this function has changed in order to produce more uniform results. In version 0.9.8 and before, this function returned two Bayes factors when nullInterval was non-NULL: the Bayes factor for the interval versus the null, and the Bayes factor for the complement of the interval versus the null. Starting in version 0.9.9, in order to get the Bayes factor for the complement, it is required to set the complement argument to TRUE, and the function only returns one Bayes factor.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>) and Jeffrey N. Rouder (<rouderj@missouri.edu>)

References

Morey, R. D. & Rouder, J. N. (2011). Bayes Factor Approaches for Testing Interval Null Hypotheses. Psychological Methods, 16, 406-419

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. Psychonomic Bulletin & Review, 16, 225-237

See Also

integrate, t.test; see ttestBF for the intended interface to this function, using the full data set.

Examples

```
## Classical example: Student's sleep data
data(sleep)
plot(extra ~ group, data = sleep)

## t.test() gives a t value of -4.0621
t.test(sleep$extra[1:10], sleep$extra[11:20], paired=TRUE)
## Gives a Bayes factor of about 15
## in favor of the alternative hypothesis
result <- ttest.tstat(t = -4.0621, n1 = 10)
exp(result[['bf']])
```

`ttestBF` *Function for Bayesian analysis of one- and two-sample designs*

Description

This function computes Bayes factors, or samples from the posterior, for one- and two-sample designs.

Usage

```
ttestBF(
  x = NULL,
  y = NULL,
  formula = NULL,
  mu = 0,
  nullInterval = NULL,
  paired = FALSE,
  data = NULL,
  rscale = "medium",
  posterior = FALSE,
  callback = function(...) as.integer(0),
  ...
)
```

Arguments

`x`	a vector of observations for the first (or only) group
`y`	a vector of observations for the second group (or condition, for paired)
`formula`	for independent-group designs, a (optional) formula describing the model
`mu`	for one-sample and paired designs, the null value of the mean (or mean difference)
`nullInterval`	optional vector of length 2 containing lower and upper bounds of an interval hypothesis to test, in standardized units
`paired`	if `TRUE`, observations are paired
`data`	for use with `formula`, a data frame containing all the data
`rscale`	prior scale. A number of preset values can be given as strings; see Details.
`posterior`	if `TRUE`, return samples from the posterior instead of Bayes factor
`callback`	callback function for third-party interfaces
`...`	further arguments to be passed to or from methods.

Details

The Bayes factor provided by `ttestBF` tests the null hypothesis that the mean (or mean difference) of a normal population is μ_0 (argument `mu`). Specifically, the Bayes factor compares two hypotheses: that the standardized effect size is 0, or that the standardized effect size is not 0. For one-sample tests, the standardized effect size is $(\mu - \mu_0)/\sigma$; for two sample tests, the standardized effect size is $(\mu_2 - \mu_1)/\sigma$.

A noninformative Jeffreys prior is placed on the variance of the normal population, while a Cauchy prior is placed on the standardized effect size. The `rscale` argument controls the scale of the prior distribution, with `rscale=1` yielding a standard Cauchy prior. See the references below for more details.

For the `rscale` argument, several named values are recognized: "medium", "wide", and "ultrawide". These correspond to r scale values of $\sqrt{2}/2$, 1, and $\sqrt{2}$ respectively.

The Bayes factor is computed via Gaussian quadrature.

Value

If `posterior` is `FALSE`, an object of class `BFBayesFactor` containing the computed model comparisons is returned. If `nullInterval` is defined, then two Bayes factors will be computed: The Bayes factor for the interval against the null hypothesis that the standardized effect is 0, and the corresponding Bayes factor for the compliment of the interval.

If `posterior` is `TRUE`, an object of class `BFmcmc`, containing MCMC samples from the posterior is returned.

Note

The default priors have changed from 1 to $\sqrt{2}/2$. The factor of $\sqrt{2}$ is to be consistent with Morey et al. (2011) and Rouder et al. (2012), and the factor of $1/2$ in both is to better scale the expected effect sizes; the previous scaling put more weight on larger effect sizes. To obtain the same Bayes factors as Rouder et al. (2009), change the prior scale to 1.

Author(s)

Richard D. Morey (<richarddmorey@gmail.com>)

References

Morey, R. D., Rouder, J. N., Pratte, M. S., & Speckman, P. L. (2011). Using MCMC chain outputs to efficiently estimate Bayes factors. Journal of Mathematical Psychology, 55, 368-378

Morey, R. D. & Rouder, J. N. (2011). Bayes Factor Approaches for Testing Interval Null Hypotheses. Psychological Methods, 16, 406-419

Rouder, J. N., Speckman, P. L., Sun, D., Morey, R. D., & Iverson, G. (2009). Bayesian t-tests for accepting and rejecting the null hypothesis. Psychonomic Bulletin & Review, 16, 225-237

See Also

`integrate`, `t.test`

Examples

```
## Sleep data from t test example
data(sleep)
plot(extra ~ group, data = sleep)

## paired t test
ttestBF(x = sleep$extra[sleep$group==1], y = sleep$extra[sleep$group==2], paired=TRUE)

## Sample from the corresponding posterior distribution
samples = ttestBF(x = sleep$extra[sleep$group==1],
           y = sleep$extra[sleep$group==2], paired=TRUE,
           posterior = TRUE, iterations = 1000)
plot(samples[,"mu"])
```

`%same%` *Compare two objects to see if they are the 'same', for some loose definition of same*

Description

Compare two objects to see if they are the 'same', for some loose definition of same

Usage

```
x %same% y
```

Arguments

x	first object
y	second object

Value

Returns `TRUE` or `FALSE`

`%termin%` *Find a model term in a vector of model terms*

Description

Find a model term in a vector of model terms

Usage

```
x %termin% table
```

Arguments

x	the terms to be matched
table	the terms to be matched against

Value

A logical vector of the same length as x, indicating if a match was located for each element of x.

Index