Package: rvalues (via r-universe)

August 31, 2024

Type Package

Title R-Values for Ranking in High-Dimensional Settings

Version 0.7.1

Date 2021-03-10

Maintainer Nicholas Henderson <nchender@umich.edu>

Imports graphics, stats, utils

Description A collection of functions for computing ``r-values'' from various kinds of user input such as MCMC output or a list of effect size estimates and associated standard errors. Given a large collection of measurement units, the r-value, r, of a particular unit is a reported percentile that may be interpreted as the smallest percentile at which the unit should be placed in the top r-fraction of units.

License GPL-2

URL https://doi.org/10.1111/rssb.12131

LazyData yes

NeedsCompilation yes

Author Nicholas Henderson [cre], Michael Newton [aut]

Date/Publication 2021-03-11 07:20:05 UTC

Repository https://nchenderson.r-universe.dev

RemoteUrl https://github.com/cran/rvalues

RemoteRef HEAD

RemoteSha 0e3b9f2528976cb0b9acae80798a5b38f57fb82d

Contents

`batavgs` *Batting Averages Data*

Description

Data set containing number of at-bats and number of hits for Major League baseball players over the 2005 season.

Usage

```
data(batavgs)
```

Format

A data frame with 929 observations on the following 7 variables.

`First.Name` factor; player's last name

`Last.Name` factor; player's first name

`Pitcher` numeric vector; an indicator of whether or not the player is a pitcher

`midseasonAB` numeric vector; number of at-bats during the first half of the season

`midseasonH` numeric vector; number of hits during the first half of the season

`TotalAB` numeric vector; total number of at-bats over the season

`TotalH` numeric vector; total number of hits over the season

Details

The 2005 Major League Baseball season was roughly six months starting from the beginning of April and ending at the beginning of October. Data from postseason play is not included. The midseason data were obtained by only considering the first three months of the season.

Source

http://projecteuclid.org/DPubS?service=UI&version=1.0&verb=Display&handle=euclid.aoas/1206367815

References

Brown, L. D. (2008), In-Season prediction of batting averages: a field test of empirical Bayes and Bayes Methodologies, *The Annals of Applied Statistics*, **2**, 1, 113–152.

Examples

```
data(batavgs)
head(batavgs)
```

`bcwest` *Breast Cancer Gene Expression Data*

Description

Effect size estimates and standard errors obtained from gene expression measurements on 7129 genes across 49 samples.

Usage

```
data(bcwest)
```

Format

A data frame with 7129 observations on the following 2 variables.

`estimates` a vector of effect size estimates

`std.err` standard errors associated with effect size estimates

Details

A description of the original data may be found in West et al. (2001). For each gene, the effect size estimate was computed from the difference in the mean expression levels of the two groups (i.e., mean(bc-positive group) - mean(bc-negative group)).

Source

T. Hothorn, P. Buehlmann, T. Kneib, M. Schmid, and B. Hofner (2013). mboost: Model-Based Boosting, R package version 2.2-3, http://CRAN.R-project.org/package=mboost.

References

Mike West, Carrie Blanchette, Holly Dressman, Erich Huang, Seiichi Ishida, Rainer Spang, Harry Zuzan, John A. Olson Jr., Jeffrey R. Marks and Joseph R. Nevins (2001), Predicting the clinical status of human breast cancer by using gene expression profiles, *Proceedings of the National Academy of Sciences*, **98**, 11462-11467.

`FDRCurve` *FDR Curve*

Description

Estimates the expected proportion of misclassified units when using a given r-value threshold. If plot=TRUE, the curve is plotted before the estimated function is returned.

Usage

```
FDRCurve(object, q, threshold = 1, plot = TRUE, xlim, ylim, xlab, ylab, main, ...)
```

Arguments

`object`	An object of class "rvals"
`q`	A value in between 0 and 1; the desired level of FDR control.
`threshold`	The r-value threshold.
`plot`	logical; if TRUE, the estimated FDR curve is plotted.
`xlim, ylim`	x and y - axis limits for the plot
`xlab, ylab`	x and y - axis labels
`main`	the title of the plot
`...`	additional arguments to `plot.default`

Details

Consider parameters of interest $(\theta_1, ..., \theta_n)$ with an effect of size of interest τ. That is, a unit is taken to be "null" if $\theta_i \leq \tau$ and taken to be "non-null" if $\theta_i > \tau$.

For r-values $r_1, ..., r_n$ and a procedure which "rejects" units satisfying $r_i \leq c$, the FDR is defined to be

$$FDR(c) = P(\theta_i < \tau, r_i \leq c)/P(r_i \leq c).$$

`FDRCurve` estimates $FDR(c)$ for values of c across (0,1) and plots (if `plot=TRUE`) the resulting curve.

Value

A list with the following two components

`fdrcurve`	A function which returns the estimated FDR for each r-value threshold.
`Rval.cut`	The largest r-value cutoff which still gives an estimated FDR less than q.

Author(s)

Nicholas Henderson and Michael Newton

See Also

OverlapCurve

Examples

```
n <- 500
theta <- rnorm(n)
ses <- sqrt(rgamma(n,shape=1,scale=1))
XX <- theta + ses*rnorm(n)
dd <- cbind(XX,ses)

rvs <- rvalues(dd, family = gaussian)

FDRCurve(rvs, q = .1, threshold = .3, cex.main = 1.5)
```

---

fluEnrich *Flu Enrichment Data*

Description

Gene-set enrichment for genes that have been identified as having an effect on influenza-virus replication.

Usage

```
data(fluEnrich)
```

Format

A data frame with 5719 observations on the following 3 variables.

`nflugenes` number of genes both annotated to the given GO term and in the collection of flu genes

`setsize` number of genes annotated to the given GO term

`GO_terms` the GO (gene ontology) term label

Details

These data were produced by associating the 984 genes (the collection of flu genes) identified in the Hao et al. (2013) meta-analysis with gene ontology (GO) gene sets (GO terms). In total, 17959 human genes were annotated to at least one GO term and 16572 GO terms were available, though this data set only contains the 5719 terms which annotated between 10 and 1000 human genes.

References

Hao, L. Q. He, Z. Wang, M. Craven, M. A. Newton, and P. Ahlquist (2013). Limited agreement of independent RNAi screens for virus-required host genes owes more to false-negatives than false-positive factors. *PLoS computational biology*, **9**, 9, e1003235.

hiv *HIV Data Set*

Description

These data contain effect size estimates and standard errors obtained from gene expression measurements on 7680 genes across 8 samples.

Usage

```
data(hiv)
```

Format

A data frame with 7680 observations on the following 2 variables.

`estimates` a vector of effect size estimates

`std.err` standard errors associated with effect size estimates

References

van't Wout, et. al. (2003), Cellular gene expression upon human immuno-deficiency virus type 1 infection of CD4+-T-Cell lines, *Journal ofVirology*, **77**, 1392–1402.

MakeGrid *Grid Construction*

Description

Computes a grid of points on the interval (0,1). This function is useful for constructing the "alpha-grid" used in various r-value computations.

Usage

```
MakeGrid(nunits, type = "log", ngrid = NULL, lower = 1/nunits, upper = 1 - lower)
```

Arguments

`nunits`	The number of units in the data for which r-values are to be calculated.
`type`	The type of grid; type can be set to `type="uniform"`, `type="log"`, or `type="log.symmetric"`.
`ngrid`	a number specifying the number of grid points
`lower`	the smallest grid point; must be greater than zero
`upper`	the largest grid point; must be less than one

Details

If `nunits` ≤ 1000, the default number of grid points is equal to `nunits`. When `nunits` > 1000, the default number of grid points is determined by

$$1000 + 25 * log(nunits - 1000) * (nunits - 1000)^{1/4}$$

.

Value

A vector of grid points in (0,1).

Author(s)

Nicholas Henderson and Michael Newton

See Also

`rvalues`

Examples

```
alpha.grid <- MakeGrid(1000,type="uniform",ngrid=200)

log.grid <- MakeGrid(40,type="log")
log.grid
hist(log.grid)
```

`MCMCtest` *MCMC sample output*

Description

a matrix of test MCMC output

Usage

```
data(MCMCtest)
```

Format

A 2000 x 400 numeric matrix. Data in the ith row should be thought of as a sample from the posterior for the ith case of interest.

See Also

`rvaluesMCMC`

mroot *Multi-dimensional Root (Zero) Finding*

Description

For a given multi-dimensional function with both a vector of lower bounds and upper bounds, `mroot` finds a vector such that each component of the function is zero.

Usage

```
mroot(f, lower, upper, ..., f.lower = f(lower, ...), f.upper = f(upper, ...),
      tol = .Machine$double.eps^0.25, maxiter = 5000)
```

Arguments

`f`	the function for which the root is sought
`lower`	a vector of lower end points
`upper`	a vector of upper end points
`...`	additional arguments to be passed to `f`
`f.lower`, `f.upper`	the same as `f(lower)` and `f(upper)`
`tol`	the convergence tolerance
`maxiter`	the maximum number of iterations

Details

The function `f` is from R^n to R^n with $f(x_1, \ldots, x_n) = (f_1(x_1), \ldots, f_n(x_n))$.

A root $x = (x_1, \ldots, x_n)$ of f satisfies $f_k(x_k) = 0$ for each component k.

`lower` $= (l_1, \ldots, l_n)$ and `upper` $= (u_1, \ldots, u_n)$ are both n-dimensional vectors such that, for each k, f_k changes sign over the interval $[l_k, u_k]$.

Value

a vector giving the estimated root of the function

Author(s)

Nicholas Henderson

See Also

`uniroot`

Examples

```
ff <- function(x,a) {
    ans <- qnorm(x) - a
    return(ans)
}
n <- 10000
a <- rnorm(n)
low <- rep(0,n)
up <- rep(1,n)

## Find the roots of ff, first using mroot and
## then by using uniroot inside a loop.

system.time(mr <- mroot(ff, lower = low, upper = up, a = a))

ur <- rep(0,n)
system.time({
for(i in 1:n) {
   ur[i] <- uniroot(ff, lower = 0, upper = 1, a = a[i])$root
}
})
```

NBA1314 *National Basketball Association, free throw data, 2013-2014 season*

Description

Free throw statistics on 482 active players, 2013-2014 season

Usage

```
data(NBA1314)
```

Format

A data frame with 482 players (rows) variables including.

`RK` rank of player by proportion of free throws made

`PLAYER` name of player

`TEAM` player's team

`GP` games played

`PPG` points per game

`FTM0` FTM/GP

`FTA0` FTA/GP

`FTA` free throws attempted

`FTM` free throws made

`FTprop` FTA/FTM

Details

Data obtained from ESPN.

References

See data analyzed in Henderson and Newton, 2015

Examples

```
data(NBA1314)
nba.dat <- cbind(NBA1314$FTM, NBA1314$FTA)
rownames(nba.dat) <- NBA1314$PLAYER

rvals.nba <- rvalues(nba.dat, family=binomial)
```

`npmixapply` *Apply Functions over estimated unit-specific posterior distributions*

Description

Using a nonparametric estimate of the mixing distribution, computes a posterior quantity of interest for each unit.

Usage

```
npmixapply(object, FUN, ...)
```

Arguments

`object`	an object of class "npmix"
`FUN`	a user provided function
`...`	optional arguments to FUN

Details

`object` is an object of class "npmix" containing a nonparametric estimate of the mixing distribution F in the following two-level sampling model $X_i|\theta_i \sim p(x|\theta_i, \eta_i)$ and $\theta_i \sim F$ for $i = 1, ..., n$.

Using `npmixapply(object, f)`, then returns the posterior expectation of f: $E[f(\theta_i)|X_i, \eta_i]$, for $i = 1, ..., n$.

Value

a vector with length equal to n

Author(s)

Nicholas Henderson

See Also

npmle

Examples

```
## Not run:
data(hiv)
npobj <- npmle(hiv, family = gaussian, maxiter = 4)

### Compute unit-specific posterior means
pmean <- npmixapply(npobj, function(x) { x })

### Compute post. prob that \theta_i < .1
pp <- npmixapply(npobj, function(x) { x < .1})

## End(Not run)
```

npmle — *Maximum Likelihood Estimate of a Mixing Distribution.*

Description

Estimates the mixture distribution nonparametrically using an EM algorithm. The estimate is discrete with the results being returned as a vector of support points and a vector of associated mixture probabilities. The available choices for the sampling distribution include: Normal, Poisson, Binomial and t-distributions.

Usage

```
npmle(data, family = gaussian, maxiter = 500, tol = 1e-4,
      smooth = TRUE, bass = 0, nmix = NULL)
```

Arguments

`data`	A data frame or a matrix with the number of rows equal to the number of sampling units. The first column should contain the main estimates, and the second column should contain the nuisance terms.
`family`	family determining the sampling distribution (see *family*)
`maxiter`	the maximum number of EM iterations
`tol`	the convergence tolerance
`smooth`	logical; whether or not to smooth the estimated cdf
`bass`	controls the smoothness level; only relevant if `smooth=TRUE`. Values of up to 10 indicate increasing smoothness.
`nmix`	optional; the number of mixture components

Details

Assuming the following two-level sampling model $X_i|\theta_i \sim p(x|\theta_i, \eta_i)$ and $\theta_i \sim F$ for $i = 1, ..., n$. The function `npmle` seeks to find an estimate of the mixing distribution F which maximizes the marginal log-likelihood

$$l(F) = \sum_i \int p(X_i|\theta, \eta_i)dF(\theta).$$

The distribution function maximizing $l(F)$ is known to be discrete; and thus, the estimated mixture distribution is returned as a set of support points and associated mixture probabilities.

Value

An object of class npmix which is a list containing at least the following components

`support`	a vector of estimated support points
`mix.prop`	a vector of estimated mixture proportions
`Fhat`	a function; obtained through interpolation of the estimated discrete cdf
`fhat`	a function; estimate of the mixture density
`loglik`	value of the log-likelihood at each iteration
`convergence`	0 indicates convergence; 1 indicates that convergence was not achieved
`numiter`	the number of EM iterations required

Author(s)

Nicholas Henderson and Michael Newton

References

Laird, N.M. (1978), Nonparametric maximum likelihood estimation of a mixing distribution, *Journal of the American Statistical Association*, **73**, 805–811.

Lindsay, B.G. (1983), The geometry of mixture likelihoods: a general theory. *The Annals of Statistics*, **11**, 86–94

See Also

`npmixapply`

Examples

```
## Not run:
data(hiv)
npobj <- npmle(hiv, family = tdist(df=6), maxiter = 25)


###  Generate Binomial data with Beta mixing distribution
n <- 3000
theta <- rbeta(n, shape1 = 2, shape2 = 10)
ntrials <- rpois(n, lambda = 10)
```

```
x <- rbinom(n, size = ntrials, prob = theta)

###  Estimate mixing distribution
dd <- cbind(x,ntrials)
npest <- npmle(dd, family = binomial, maxiter = 25)

### compare with true mixture cdf
tt <- seq(1e-4,1 - 1e-4, by = .001)
plot(npest, lwd = 2)
lines(tt, pbeta(tt, shape1 = 2, shape2 = 10), lwd = 2, lty = 2)

## End(Not run)
```

`OverlapCurve` *Overlap Curve*

Description

Estimates the expected proportion of units in the top fraction and those deemed to be in the top fraction by the r-value procedure. If plot=TRUE, the curve is plotted before the estimated function is returned.

Usage

```
OverlapCurve(object, plot = TRUE, xlim, ylim, xlab, ylab, main, ...)
```

Arguments

`object`	An object of class "rvals"
`plot`	logical. If TRUE, the estimated overlap curve is plotted.
`xlim`, `ylim`	x and y - axis limits for the plot
`xlab`, `ylab`	x and y - axis labels
`main`	the title of the plot
`...`	additional arguments to `plot.default`

Details

For parameters of interest $\theta_1, ..., \theta_n$ and corresponding r-values $r_1, ..., r_n$, the overlap at a particular value of α is defined to be

$$overlap(\alpha) = P(\theta_i \geq \theta_\alpha, r_i \leq \alpha),$$

where the threshold θ_α is the upper-αth quantile of the distribution of the θ_i (i.e., $P(\theta_i \geq \theta_\alpha) = \alpha$). `OverlapCurve` estimates this overlap for values of alpha across (0,1) and plots (if `plot=TRUE`) the resulting curve.

Value

A function returning estimated overlap values.

Author(s)

Nicholas Henderson and Michael Newton

References

Henderson, N.C. and Newton, M.A. (2016). *Making the cut: improved ranking and selection for large-scale inference*. J. Royal Statist. Soc. B., 78(4), 781-804. doi: 10.1111/rssb.12131 `https://arxiv.org/abs/1312.5776`

Examples

```
n <- 500
theta <- rnorm(n)
ses <- sqrt(rgamma(n,shape=1,scale=1))
XX <- theta + ses*rnorm(n)
dd <- cbind(XX,ses)

rvs <- rvalues(dd, family = gaussian)

OverlapCurve(rvs, cex.main = 1.5)
```

PostPercentile *Posterior expected percentiles*

Description

Computes posterior expected percentiles for both parametric and nonparametric models.

Usage

```
PostPercentile(object)
```

Arguments

`object`	An object of class "rvals"

Details

With parameters of interest $\theta_1, ..., \theta_n$ the rank of the ith parameter (when we set the ranking so that the largest θ_i gets rank 1) is defined as $rank(\theta_i) = sum_j(\theta_j \geq \theta_i)$ and the associated percentile is $perc(\theta_i) = rank(\theta_i)/(n + 1)$. The posterior expected percentile for the ith unit (see e.g., Lin et. al. (2006)) is simply the expected value of $perc(\theta_i)$ given the data.

The function `PostPercentile` computes an asymptotic version of the posterior expected percentile, which is defined as

$$P(\theta_i \leq \theta | data),$$

where θ has the same distribution as θ_i and is independent of both θ_i and the data. See Henderson and Newton (2014) for additional details.

Value

A vector of estimated posterior expected percentiles.

Author(s)

Nicholas Henderson and Michael Newton

References

Henderson, N.C. and Newton, M.A. (2016). *Making the cut: improved ranking and selection for large-scale inference*. J. Royal Statist. Soc. B., 78(4), 781-804. doi: 10.1111/rssb.12131 `https://arxiv.org/abs/1312.5776`

Lin, R., Louis, T.A., Paddock, S.M., and Ridgeway, G. (2006). Loss function based ranking in two-stage, hierarchical models. *Bayesian Analysis*, **1**, 915–946.

See Also

`rvalues`

Examples

```
n <- 3000
theta <- rnorm(n, sd = 3)
ses <- sqrt(rgamma(n, shape = 1, scale = 1))
XX <- theta + ses*rnorm(n)
dd <- cbind(XX,ses)

rv <- rvalues(dd, family = gaussian)

perc <- PostPercentile(rv)
plot(rv$rvalues, perc)
```

`PostSummaries` *R-values from posterior summary quantities*

Description

Computes r-values assuming that, for each parameter of interest, the user supplies a value for the posterior mean and the posterior standard deviation. The assumption here is that the posterior distributions are Normal.

Usage

```
PostSummaries(post.means, post.sds, hypers = NULL, qtheta = NULL, alpha.grid = NULL,
                ngrid = NULL, smooth = 0)
```

Arguments

`post.means` a vector of posterior means

`post.sds` a vector of posterior standard deviations

`hypers` a list with two elements: mean and sd. These represent the parameters in the (Normal) prior which was used to generate the posterior means and sds. If hypers is not supplied then one must supply the quantile function qtheta.

`qtheta` a function which returns the quantiles (for upper tail probs.) of theta. If this is not supplied, the hyperparameter must be supplied.

`alpha.grid` grid of values in (0,1); used for the discrete approximation approach for computing r-values.

`ngrid` number of grid points for alpha.grid; only relevant when `alpha.grid = NULL`

`smooth` either `smooth="none"` or `smooth` takes a value between 0 and 10; this determines the level of smoothing applied to the estimate of $\lambda(\alpha)$; if `smooth` is given a number, the number is used as the `bass` argument in `supsmu`.

Value

An object of class "rvals"

Author(s)

Nicholas Henderson and Michael Newton

See Also

`rvalues`

Examples

```
n <- 500
theta <- rnorm(n)
sig_sq <- rgamma(n,shape=1,scale=1)
X <- theta + sqrt(sig_sq)*rnorm(n)

pm <- X/(sig_sq + 1)
psd <- sqrt(sig_sq/(sig_sq + 1))

rvs <- PostSummaries(post.means=pm,post.sds=psd,hypers=list(mean=0,sd=1))
hist(rvs$rvalues)
```

rvalueBoot *Bootstrapped r-values*

Description

Estimates a new prior for each bootstrap replications ... (need to add)

Usage

```
rvalueBoot(object, statistic = median, R, type = "nonparametric")
```

Arguments

`object`	An object of class "rvals"
`statistic`	The statistic used to summarize the bootstrap replicates.
`R`	Number of bootstrap replicates
`type`	Either `type="nonparametric"` or `type="parametric"`; the nonparametric type corresponds to the usual bootstrap where units are sampled with replacement.

Details

When `type="nonparametric"`, the prior is re-estimated (using the resampled data) in each bootstrap replication, and r-values are re-computed with respect to this new model.

When `type="parametric"`,

Value

A list with the following two components

`rval.repmat`	A matrix where each column corresponds to a separate bootstrap replication.
`rval.boot`	A vector of r-values obtained by applying the statistic to each row of `rval.repmat`.

Author(s)

Nicholas Henderson and Michael Newton

References

Henderson, N.C. and Newton, M.A. (2016). *Making the cut: improved ranking and selection for large-scale inference*. J. Royal Statist. Soc. B., 78(4), 781-804. doi: 10.1111/rssb.12131 `https://arxiv.org/abs/1312.5776`

See Also

`rvalues`

Examples

```
## Not run:
n <- 3000
theta <- rnorm(n, sd = 3)
ses <- sqrt(rgamma(n, shape = 10, rate = 1))
XX <- theta + ses*rnorm(n)
dd <- cbind(XX,ses)

rv <- rvalues(dd, family = gaussian, prior = "conjugate")

rvb <- rvalueBoot(rv, R = 10)
summary(rvb$rval.repmat[512,])

## End(Not run)
```

rvalues *R-values*

Description

Given data on a collection of units, this function computes r-values which are percentiles constructed to maximize the agreement between the reported percentiles and the percentiles of the effect of interest. Additional details about r-values are provided below and can also be found in the listed references.

Usage

```
rvalues(data, family = gaussian, hypers = "estimate", prior = "conjugate",
        alpha.grid = NULL, ngrid = NULL, smooth = "none", control = list())
```

Arguments

`data` A data frame or a matrix with the number of rows equal to the number of sampling units. The first column should contain the main estimates, and the second column should contain the nuisance terms.

`family` An argument which determines the sampling distribution; this could be either `family = gaussian`, `family = tdist`, `family = binomial`, `family = poisson`

`hypers` values of the hyperparameters; only meaningful when the conjugate prior is used; if set to "estimate", the hyperparameters are found through maximum likelihood; if not set to "estimate" the user should supply a vector of length two.

`prior` the form of the prior; either `prior="conjugate"` or `prior="nonparametric"`.

`alpha.grid` a numeric vector of points in (0,1); this grid is used in the discrete approximation of r-values

`ngrid` number of grid points for alpha.grid; only relevant when `alpha.grid=NULL` `smooth` either `smooth="none"` or `smooth` takes a value between 0 and 10; this determines the level of smoothing applied to the estimate of $\lambda(\alpha)$ (see below for the definition of $\lambda(\alpha)$); if `smooth` is given a number, the number is used as the `bass` argument in `supsmu`.

`control` a list of control parameters for estimation of the prior; only used when the prior is nonparametric

Details

The r-value computation assumes the following two-level sampling model $X_i|\theta_i \sim p(x|\theta_i, \eta_i)$ and $\theta_i \sim F$, for $i = 1, ..., n$, with parameters of interest θ_i, effect size estimates X_i, and nuisance terms η_i. The form of $p(x|\theta_i, \eta_i)$ is determined by the `family` argument. When `family = gaussian`, it is assumed that $X_i|\theta_i, \eta_i \sim$ N(θ_i, η_i^2). When `family = binomial`, the (X_i, η_i) represent the number of successes and number of trials respectively, and it is assumed that $X_i|\theta_i, \eta_i \sim$ Binomial(θ_i, η_i). When `family = poisson`, the X_i should be counts, and it is assumed that $X_i|\theta_i, \eta_i \sim$ Poisson$(\theta_i * \eta_i)$.

The distribution of the effect sizes F may be a parametric distribution that is conjugate to the corresponding `family` argument, or it may be estimated nonparametrically. When it is desired that F be parametric (i.e., `prior = "conjugate"`), the default is to estimate the hyperparameters (i.e., `hypers = "estimate"`), but these may be supplied by the user as a vector of length two. To estimate F nonparametrically, one should use `prior = "nonparametric"` (see `npmle` for further details about nonparametric estimation of F).

The *r-value*, r_i, assigned to the ith case of interest is determined by r_i = inf[$0 < \alpha < 1$: $V_\alpha(X_i, \eta_i) \geq \lambda(\alpha)$] where $V_\alpha(X_i, \eta_i) = P(\theta_i \geq \theta_\alpha | X_i, \eta_i)$ is the posterior probability that θ_i exceeds the threshold θ_α, and $\lambda(\alpha)$ is the upper-αth quantile associated with the marginal distribution of $V_\alpha(X_i, \eta_i)$ (i.e., $P(V_\alpha(X_i, \eta_i) \geq \lambda(\alpha)) = \alpha$). Similarly, the threshold θ_α is the upper-αth quantile of F (i.e., $P(\theta_i \geq \theta_\alpha) = \alpha$).

Value

An object of class "rvals" which is a list containing at least the following components:

`main` a data frame containing the r-values, the r-value rankings along with the rankings from several other common procedures

`aux` a list containing other extraneous information

`rvalues` a vector of r-values

Author(s)

Nicholas C. Henderson and Michael A. Newton

References

Henderson, N.C. and Newton, M.A. (2016). *Making the cut: improved ranking and selection for large-scale inference*. J. Royal Statist. Soc. B., 78(4), 781-804. doi: 10.1111/rssb.12131 `https://arxiv.org/abs/1312.5776`

See Also

rvaluesMCMC, PostSummaries, Valpha

Examples

```
## Not run:
### Binomial example with Beta prior:
data(fluEnrich)
flu.rvals <- rvalues(fluEnrich, family = binomial)
hist(flu.rvals$rvalues)

### look at the r-values for indices 10 and 2484
fig_indices  <- c(10,2484)
fluEnrich[fig_indices,]

flu.rvals$rvalues[fig_indices]

### Gaussian sampling distribution with nonparametric prior
### Use a maximum of 5 iterations for the nonparam. estimate
data(hiv)
hiv.rvals <- rvalues(hiv, prior = "nonparametric")

## End(Not run)
```

---

rvaluesMCMC *R-values from MCMC output.*

Description

Returns r-values from an array of MCMC output.

Usage

```
rvaluesMCMC(output, qtheta, alpha.grid = NULL, ngrid = NULL, smooth = "none")
```

Arguments

output	a matrix contatining mcmc ouput. The ith row should represent a sample from the posterior of the ith parameter of interest.
qtheta	either a function which returns the quantiles (for upper tail probs.) of theta or a vector of theta-quantiles.
alpha.grid	grid of values in (0,1); used for the discrete approximation approach for computing r-values.
ngrid	number of grid points for alpha.grid; only relevant when alpha.grid = NULL
smooth	either smooth="none" or smooth takes a value between 0 and 10; this determines the level of smoothing applied to the estimate of $\lambda(\alpha)$; if smooth is given a number, the number is used as the bass argument in supsmu.

Value

An object of class "rvals" which is a list containing at least the following components:

`main`	a data frame containing the r-values, the r-value rankings along with the rankings from several other common procedures
`aux`	a list containing other extraneous information
`rvalues`	a vector of r-values

Author(s)

Nicholas Henderson and Michael Newton

References

Henderson, N.C. and Newton, M.A. (2016). *Making the cut: improved ranking and selection for large-scale inference*. J. Royal Statist. Soc. B., 78(4), 781-804. doi: 10.1111/rssb.12131 `https://arxiv.org/abs/1312.5776`

See Also

`rvalues`, `PostSummaries`

Examples

```
data(MCMCtest)

### For the MCMC output in MCMC_test, the prior assumed for the effect sizes of
### interest was a mixture of two t-distributions. The function qthetaTMix
### computes the quantiles for this prior.

qthetaTMix <- function(p) {
    ### function to compute quantiles (for upper tail probabilities) for a
    ###  mixture of two t-distributions
    mu <- c(.35,-.12)
    sig <- c(.2,.08)
    mix.prop <- c(.25,.75)

    ff <- function(x,pp) {
        prob_less <- 0
        for(k in 1:2) {
         prob_less <- prob_less + pt((x - mu[k])/sig[k],df=4,lower.tail=FALSE)*mix.prop[k]
        }
        return(prob_less - pp)
    }

    nn <- length(p)
    ans <- numeric(nn)
    for(i in 1:nn) {
        ans[i] <- uniroot(ff,interval=c(-5,5),tol=1e-6,pp=p[i])$root
    }
    return(ans)
}

rvs <- rvaluesMCMC(MCMCtest, qtheta = qthetaTMix)
```

tdist *t-distribution family object*

Description

A t-distribution family object which allows one to specify a t-density for the sampling distribution. Modeled after `family` objects often used in the glm function.

Usage

```
tdist(df)
```

Arguments

`df`	vector containing the degrees of freedom

Value

An object of class `"newfam"`, which is a list containing the following components

`family`	The family name
`df`	The degrees of freedom

Author(s)

Nicholas Henderson and Michael Newton

See Also

`family`, `glm`, `npmle`

Examples

```
a <- tdist(df=5)
```

TopList *List of Top Units*

Description

Returns a list of the top units ranked according to "r-value" or another specified statistic.

Usage

```
TopList(object, topnum = 10, sorted.by = c("RValue","PostMean","MLE","PVal"))
```

Arguments

`object`	An object of class "rvals"
`topnum`	The length of the top list.
`sorted.by`	The statistic by which to sort; this could be `sorted.by = "RValue"`, `sorted.by = "PostMean"`, `sorted.by = "MLE"`, or `sorted.by = "PVal"`

Value

a data frame with `topnum` rows and columns containing the r-value, mle, posterior mean, and p-value rankings.

Author(s)

Nicholas Henderson and Michael Newton

See Also

`rvalues`

Examples

```
n <- 500
theta <- rnorm(n)
ses <- sqrt(rgamma(n,shape=1,scale=1))
XX <- theta + ses*rnorm(n)
dd <- cbind(XX,ses)

rvs <- rvalues(dd, family = gaussian)

TopList(rvs, topnum = 12)
TopList(rvs, topnum = 15, sorted.by = "MLE")
```

Valpha *R-values from a matrix of posterior tail probabilities.*

Description

Computes r-values directly from a "Valpha" matrix V where each column of Valpha contains posterior tail probabilities relative to a threshold indexed by alpha.

Usage

```
Valpha(V, alpha.grid, smooth = "none")
```

Arguments

`V` a numeric vector with (i,j) entry: V[i,j] = P(theta_i >= theta[alpha_j]|data)

`alpha.grid` grid of values in (0,1); used for the discrete approximation approach for computing r-values.

`smooth` either `smooth="none"` or `smooth` takes a value between 0 and 10; this determines the level of smoothing applied to the estimate of $\lambda(\alpha)$; if `smooth` is given a number, the number is used as the `bass` argument in `supsmu`.

Value

A list with the following components

`rvalues` a vector of computed r-values

`Vmarginals` The estimated V-marginals along the alpha grid points

`Vmarginals.smooth` a function obtained through interpolation and smoothing (if desired) the Vmarginals; i.e., an estimate of $\lambda(\alpha)$ (see `rvalues`)

Author(s)

Nicholas Henderson and Michael Newton

References

Henderson, N.C. and Newton, M.A. (2016). *Making the cut: improved ranking and selection for large-scale inference*. J. Royal Statist. Soc. B., 78(4), 781-804. doi: 10.1111/rssb.12131 `https://arxiv.org/abs/1312.5776`

See Also

`rvalues` `rvaluesMCMC`

Examples

```
## Not run:
data(fluEnrich)
rvobj <- rvalues(fluEnrich, family = binomial)

Vrvals <- Valpha(rvobj$aux$V, rvobj$aux$alpha.grid)

## End(Not run)
```

Index