Package 'metahdep'

November 14, 2025

Type Package

Title Hierarchical Dependence in Meta-Analysis

Version 1.68.0

Date 2011-09-15

Author John R. Stevens, Gabriel Nicholas

Maintainer John R. Stevens <john.r.stevens@usu.edu>

Depends R (>= 2.10), methods

Suggests affyPLM

Description Tools for meta-analysis in the presence of hierarchical (and/or sampling) dependence, including with gene expression studies

Collate ES.obj.R metaprep.R getPLM.es.R metahdep.format.R metahdep.R metahdep.FEMA.R metahdep.REMA.R metahdep.HBLM.R metahdep.other.R

License GPL-3

biocViews Microarray, DifferentialExpression

git_url https://git.bioconductor.org/packages/metahdep

git_branch RELEASE_3_22

git_last_commit d46b885

git_last_commit_date 2025-10-29

Repository Bioconductor 3.22

Date/Publication 2025-11-13

Contents

ES.obj-class 2
getPLM.es 3
gloss 4
HGU.DifExp.list 5
HGU.newnames 6
HGU.prep.list 7
metahdep 8
metahdep.FEMA 10
metahdep.format 12

metahdep.HBLM 13
metahdep.other 16
metahdep.REMA 17
metaprep-class 19

Index **21**

`ES.obj-class` *Class ES.obj*

Description

This is a class representation for the effect size estimates and other summary information from a single gene expression study, usually constructed in preparation for meta-analysis.

Objects from the Class

Objects can be created using the functions `getPLM.es` or `new`.

Slots

`gn`**:** Object of class `character` representing the probeset IDs of the genes in the study.

`ES.mat`**:** Object of class `matrix` representing the effect size estimates for each gene in the study. Rows correspond to probesets and columns correspond to different comparisons or tests of differential expression. If a test of differential expression was performed for different covariate levels, then there will be more than one column, so that each row in this matrix represents a vector of effect size estimates for the corresponding probeset in the `gn` slot.

`Cov.mat`**:** Object of class `matrix`, with each row representing the upper triangle of the variance / covariance matrix of the vector of effect size estimates (row in the `ES.mat` slot) for the corresponding probeset in the `gn` slot. Within each row, the order is the same as the result of a call to the `upperTriangle(matrix,diag=T)` function in the *gdata* package.

`chip`**:** Object of class `character` representing the chip or array version used in the study.

`covariates`**:** Object of class `data.frame` representing covariate differences among the columns of the matrix in the `ES.mat` slot. This object has a row for each column of the matrix in the `ES.mat` slot, and a column for each covariate to be considered in the meta-analysis, regardless of whether the covariate takes on multiple values in the study represented in this `ES.obj` object. For best interpretability, columns of the `data.frame` in this `covariates` slot should be coded as 0/1.

`dep.grp` Object of class `integer` representing the dependence group number assigned to the study. Studies from the same research team may be considered hierarchically dependendent and share the same `dep.grp` value.

Methods

@ replace the slot entries

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
###
###  See the metahdep package vignette for a full example
###
data(HGU.DifExp.list)
ES <- HGU.DifExp.list[[1]]
slotNames(ES)
head(ES@gn)
head(ES@ES.mat)
head(ES@Cov.mat)
ES@chip
ES@covariates
ES@dep.grp
```

---

getPLM.es *getPLM.es*

Description

Calculates effect size estimates for a single study, based on a probe-level model, in preparation for a meta-analysis. It returns an `ES.obj` object containing the result.

Usage

```
getPLM.es(abatch, trt1, trt2, covariates=NULL, dep.grp=NULL,
          sub.gn=NULL, bg.norm=TRUE)
```

Arguments

`abatch`	An `AffyBatch` object containing the data of interest.
`trt1`	A vector (or list of vectors) of array indices for treatment level 1 (control). If more than one test of differential expression is to be performed (for multiple covariate levels, for example), this should be a list of vectors; each `trt1` / `trt2` vector pair defines a comparison of interest.
`trt2`	A vector (or list of vectors) of array indices for treatment level 2 (treatment). If more than one test of differential expression is to be performed (for multiple covariate levels, for example), this should be a list of vectors; each `trt1` / `trt2` vector pair defines a comparison of interest.
`covariates`	(optional) A `data.frame` object representing covariate differences, if any, among the comparisons defined by `trt1` / `trt2` vector pairs. This `data.frame` should have a named column for each covariate to be considered in the meta-analysis, regardless of whether the covariate takes on multiple values in the study represented by the `abatch` argument. This `data.frame` must have a row for each comparison of interest, as defined by the `trt1` / `trt2` vector pairs. Elements of this `data.frame` should be coded numerically.
`dep.grp`	(optional) A single numeric value representing the dependence group number assigned to the study. Studies from the same research team may be considered hierarchically dependendent and share the same value.

sub.gn	(optional) A vector of geneNames (probe set ID's); the probe-level model will only be fit for these probesets. If NULL (default), all probesets are used.
bg.norm	(optional) A logical value specifying whether or not to perform background correction and normalization before fitting the probe-level model.

Details

For some subset of probesets in a gene expression study, this function calculates the effect size estimates based on Bolstad's probe-level model (Bolstad 2004), as described in Hu et al. (2006). Only two-group comparisons (treatment vs. control, for example) are supported. This is done in preparation for a meta-analysis of multiple gene expression studies.

Value

An object of class ES.obj

Author(s)

John R. Stevens, Gabriel Nicholas

References

Bolstad B. M. (2004), *Low-level Analysis of High-density Oligonucleotide Array Data: Background, Normalization and Summarization*, PhD dissertation, U.C. Berkeley.

Hu P., Greenwood C.M.T., and Beyene J. (2006), Integrative Analysis of Gene Expression Data Including an Assessment of Pathway Enrichment for Predicting Prostate Cancer, *Cancer Informatics* 2006:2 289-300.

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
###
###  See the metahdep package vignette for a full example
###
```

---

gloss *gloss: Data from the JEBS glossing paper*

Description

This includes the following four objects:

gloss.Table1	a data.frame containing the contents of Table 1 in the paper (sample sizes, sample means, sample SDs, and covariate information from each study)
gloss.X	a matrix representing the design matrix X for the meta-analysis in the paper
gloss.theta	a vector, representing the effect size estimates as summarized in Table 3 of the paper

`gloss.V` a matrix, representing the covariance matrix of effect size estimates, including sampling dependence in off-diagonal elements, as summarized in Table 3 of paper

Usage

```
data(gloss)
```

Format

This object contains the four objects described above.

Details

This data set summarizes 13 experiments with 18 study reports, all involving the effect of native-language (L1) vocabulary aids on second language (L2) reading comprehension. Some experiments produced multiple study reports, creating a dependence structure among the resulting effect size estimates.

The intended use for these data is to demonstrate the methods coded in the *metahdep* package.

References

Stevens J.R. and Taylor A.M. (2009), Hierarchical Dependence in Meta-Analysis, *Journal of Educational and Behavioral Statistics*, 34(1):46-73.

See also the *metahdep* package vignette.

Examples

```
data(gloss)
# Look at Table 1
gloss.Table1
```

HGU.DifExp.list *HGU.DifExp.list: A list of 4 ES.obj objects*

Description

An illustrative example of a list summarizing several studies of gene expression. This is used as an example in the meta-analyis of hierarchically dependent gene expression studies.

Usage

```
data(HGU.DifExp.list)
```

Format

This object is a list containing 4 `ES.obj` objects. Each `ES.obj` object represents the results from a separate gene expression study.

Details

This object has been assembled from existing data as an artificial example; see the vignette for details on its construction. In this example, four studies were conducted, and can be summarized as follows:

Study	Lab	Tissue	Chip
1	1	0	hgu133a
1	1	1	hgu133a
2	1	0	hgu95a
3	2	0	hgu95av2
4	3	1	hgu133b

Notice that study 1 involved two tissue types. The vignette shows how this example supposes that sampling dependence was introduced in study 1 by fitting a gene-specific model with both tissue types simultaneously. Hierarchical dependence is also present in these data because studies 1 and 2 were conducted by the same lab. Each element of `HGU.DifExp.list` is an `ES.obj` object in the same format as returned by the `getPLM.es()` function. Look at the elements of the list (and the vignette) to get an idea of how the data should be laid out.

The intended use for these data is to demonstrate a meta-analysis procedure that accounts for hierarchical dependence between studies. The idea is that results from different studies from the same lab might be dependent. This is an example object that is to be passed as an argument to the `metahdep.format()` function.

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
data(HGU.DifExp.list)
head(HGU.DifExp.list[[1]]@ES.mat)
HGU.DifExp.list[[1]]@covariates
##  etc.
```

`HGU.newnames` *HGU.newnames*

Description

An illustrative example of a `data.frame` with 3 columns. The first column is named "chip", the second "old.name", and the third "new.name". The rows each hold the name of a chip type, a chip-specific probeset name, and a common name used to match probesets across different chip versions.

Usage

```
data(HGU.newnames)
```

Format

A data.frame with observations on the following 3 variables for a subset of probesets on different chip types.

chip a character vector specifying the chip type

old.name a character vector specifying the probeset name on the chip type

new.name a character vector specifying the common identifier, such as an Entrez Gene ID, for the probeset on the chip type

Details

This is an example of a newnames argument that is required by the metahdep.format function. When paired with a list of ES.obj class objects (see HGU.DifExp.list) this allows the metahdep.format() function to assemble all of the information from all of the studies for a specific gene. The new.name is a 'common' identifier, e.g., an Entrez Gene ID. Different studies may use different chip types, or different versions of chips, where information for a gene with a particular Entrez Gene ID may have a different probeset name on each chip type. This newnames argument is meant to facilitate the matching of gene information across different chip types. See the *metahdep* package vignette for more details on the construction of this object.

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
data(HGU.newnames)
head(HGU.newnames)
```

HGU.prep.list *HGU.prep.list*

Description

An illustrative example of a list, where each element is a metaprep class object with data for a particular gene. It comes from the study data held within the HGU.DifExp.list object.

Usage

```
data(HGU.prep.list)
```

Format

This object is a list of metaprep objects.

Details

Each element of HGU.prep.list is a gene-specific object of class metaprep.

HGU.prep.list is an example of an object created by the metahdep.format() function; see the help file for the metahdep.format() function and the *metahdep* package vignette for details on its construction. HGU.prep.list is an example object used as an argument to the metahdep() function. The individual elements of this object can be extracted and passed as arguments to the more general meta-analysis functions, metahdep.HBLM(), metahdep.REMA(), and metahdep.FEMA().

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
data(HGU.prep.list)

HGU.prep.list[[7]]
## etc...
```

---

metahdep *metahdep*

Description

Takes a list of metaprep objects and performs the specified meta-analysis on each element. Intended mainly for meta-analyzing the results of multiple gene expression studies.

Usage

```
metahdep(prep.list, genelist = NULL, method = "HBLM", n = 10, m = 10,
         center.X = FALSE, delta.split = FALSE, return.list = FALSE,
         two.sided = TRUE)
```

Arguments

prep.list	A list of metaprep class objects as returned by the metahdep.format() function.
genelist	(optional) A subsetting parameter. A vector of gene/probeset names on which to perform the meta-analyses.
method	(optional) One of: "FEMA" - fixed effects meta-analysis, "REMA" - random effects meta-analysis, or "HBLM" - hierarchical Bayes linear model. This defaults to "HBLM".
n	(optional) An even integer specifying the number of steps to take over each quartile in the numerical integration over tau when doing HBLM. See metahdep.HBLM.
m	(optional) An even integer specifying the number of steps to take in the numerical integration over varsigma (given tau) when doing HBLM. See metahdep.HBLM.

`center.X` (optional) A logical value specifying whether or not to center the columns of the covariate matrices. If `TRUE`, then for the covariate matrix of each `metaprep` object, the mean each non-intercept column will be subtracted from every element in the column prior to the meta-analysis. This changes the interpretation of the intercept coefficient estimate from the model fit.

`delta.split` (optional) A logical value specifying whether or not to account for hierarchical dependence via delta-splitting. Only used in methods "REMA" and "HBLM". If `TRUE`, then each `metaprep` object must include a dependence matrix M.

`return.list` (optional) A logical value specifying whether to return the results as a list of lists rather than as a `data.frame`. The default is `FALSE`.

`two.sided` (optional) A logical value specifying whether to transform the posterior probabilities from the HBLM method. The default `TRUE` returns 2-sided p-values for the parameter estimates for convenience in interpretation. If this is set to FALSE, then it will return 1-sided posterior probabilities representing P(beta[j] > 0 | data).

Value

Returns a `data.frame` by default. The exact contents of the `data.frame` will vary depending on the `method` argument. The row names of the `data.frame` will be the gene names from the `prep.list` argument. For all `method` options, the first several columns of the resulting `data.frame` will be the model parameter estimates (beta hats). The next group of columns will be the elements of the variance/covariance matrix for the beta hats. The next group of columns will be the p-values for the parameter estimates. The remaining columns will change depending on the method.

For FEMA (and REMA), the remaining columns are the Q statistic and its p-value – testing for model homogeneity.

For HBLM, the remaining columns are the posterior mean and variance of tau, the posterior mean and variance of varsigma, and the posterior covariance of tau and varsigma.

All columns in the `data.frame` have meaningful names to aid their interpretation.

Author(s)

John R. Stevens, Gabriel Nicholas

References

Stevens J.R. and Doerge R.W. (2005), A Bayesian and Covariate Approach to Combine Results from Multiple Microarray Studies, *Proceedings of Conference on Applied Statistics in Agriculture*, pp. 133-147.

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

Stevens J.R. and Taylor A.M. (2009), Hierarchical Dependence in Meta-Analysis, *Journal of Educational and Behavioral Statistics*, 34(1):46-73.

See also the *metahdep* package vignette.

Examples

```
data(HGU.prep.list)

##  do FEMA and REMA, and view the results
FEMA.results <- metahdep(HGU.prep.list, method="FEMA", center.X=TRUE)
```

```
head(FEMA.results)

REMA.results <- metahdep(HGU.prep.list, method="REMA", center.X=TRUE)
head(REMA.results)

##  get a small subset of genes
##  some of these may not be suitable for all methods
##  (there may not be enough data for that gene)
data(HGU.newnames)
set.seed(123)
gene.subset <- sample(HGU.newnames$new.name, 50)

##  view results from REMA and HBLM with delta splitting on subset of genes
REMA.dsplit.results <- metahdep(HGU.prep.list, method="REMA",
  genelist=gene.subset, delta.split=TRUE, center.X=TRUE)
head(REMA.dsplit.results)

HBLM.dsplit.results <- metahdep(HGU.prep.list, method="HBLM",
  genelist=gene.subset, delta.split=TRUE, center.X=TRUE)
head(HBLM.dsplit.results)
```

`metahdep.FEMA` *metahdep.FEMA*

Description

Performs a fixed effects linear model meta-analysis. It returns a list containing the results.

Usage

```
metahdep.FEMA(theta, V, X, meta.name = "meta-analysis",
              center.X = FALSE)
```

Arguments

`theta`	A vector of effect size estimates from multiple studies.
`V`	The variance/covariance matrix for `theta`. Typically, this will be block diagonal (to represent any sampling dependence).
`X`	A matrix of covariates for `theta`. At the very least, this must consist of an intercept term. Other covariates can be included, but there must be more rows than columns in this covariate matrix.
`meta.name`	(optional) A name field for bookkeeping. This can be any character string.
`center.X`	(optional) A logical value specifying whether or not to center the columns of `X`. If `TRUE`, then the mean from each column will be subtracted from every element in that column (but not for the intercept). This changes the interpretation of the intercept coefficient estimate from the model fit.

Details

Takes a vector of effect size estimates, a variance/covariance matrix, and a covariate matrix, and fits a fixed effects linear model meta-analysis. When a meta-analysis is to be performed for gene expression data (on a per-gene basis), the `metahdep()` function calls this function for each gene separately.

Value

A list with the following named components:

`beta.hats`	A vector of model estimates for the covariates given by `X` (it may be a scalar, i.e., a vector of length 1)
`cov.matrix`	The variance/covariance matrix for the `beta.hats` estimate(s)
`beta.hat.p.values`	The [two-sided] p-value(s) for the `beta.hats` estimate(s)
`Q`	The statistic used to test for model homogeneity / model mis-specification
`Q.p.value`	The p-value for `Q`
`name`	An optional name field

Author(s)

John R. Stevens, Gabriel Nicholas

References

Hedges L. V. and Olkin I (1985), *Statistical methods for meta-analysis*, San Diego, CA: Academic Press.

Stevens J.R. and Doerge R.W. (2005), Combining Affymetrix Microarray Results, *BMC Bioinformatics* 6:57.

Stevens J.R. and Taylor A.M. (2009), Hierarchical Dependence in Meta-Analysis, *Journal of Educational and Behavioral Statistics*, 34(1):46-73.

See also the *metahdep* package vignette.

Examples

```
###
### Example 1: gene expression data
### - this uses one gene from the HGU.prep.list object

# load data and extract components for meta-analysis (for one gene)
data(HGU.prep.list)
gene.data <- HGU.prep.list[[7]]
theta <- gene.data@theta
V <- gene.data@V
X <- gene.data@X
gene.name <- gene.data@gene

# fit a regular FEMA (no hierarchical dependence)
results <- metahdep.FEMA(theta, V, X, meta.name=gene.name, center.X=TRUE)
results

###
###  Example 2: glossing data
###  - this produces part of Table 5 in the Stevens and Taylor JEBS paper.

data(gloss)
FEMA <- metahdep.FEMA(gloss.theta, gloss.V, gloss.X, center.X=TRUE)
round(cbind( t(FEMA$beta.hats), t(FEMA$beta.hat.p.values)),4)
```

metahdep.format *metahdep.format*

Description

This function is intended to facilitate the meta-analysis of multiple gene expression studies. This function takes the results from a number of studies and collects together the data for each gene, in preparation for a meta-analysis.

Usage

```
metahdep.format(ES.obj.list, newnames, min.var = 0.0001,
                include.row.indices = FALSE, show.warnings = FALSE,
                pd.verify = FALSE)
```

Arguments

ES.obj.list	A list object containing the results from multiple gene expression studies. Each element of the list is an object of class ES.obj.
newnames	A data.frame object that describes how to merge the results from different studies. It must have 3 columns, and a row for each probeset to be considered. The first column must be named "chip" and contain a name of a chip version used in one of the studies, corresponding to the "chip" slots of the ES.obj.list argument. The second column must be named "old.name" and must hold the chip-specific name for each probeset. The third column must be named "new.name" and should hold the general name (Entrez ID, for example) that can be used to identify common probesets on the different chip versions. See the HGU.newnames object in the given example code (and its creation in the vignette) for guidance on how to build this object.
min.var	(optional) A positive real number that acts as a lower bound on the allowed variances for any measure of differential expression. This might be used to guard against specious claims of significance due to naturally low variance.
include.row.indices	(optional) A logical value to determine whether or not to include the study and row indices for the data for each gene in the returned list of metaprep objects. Only used for debugging purposes, and so defaults to FALSE.
show.warnings	(optional) A logical value to determine whether or not to display warnings in certain situations, like if a gene is expected in a particular study but not found, or if a gene is not found in any study, or on some other instances. This may sometimes cause a large number of uninteresting warnings to be displayed, and so defaults to FALSE.
pd.verify	(optional) A logical value to determine whether or not to check the generated variance/covariance matrix for positive-definiteness. Since this should always be the case, this would indicate a problem somewhere in the data – more specifically, in the covariance values of one of the studies in the ES.obj.list argument. Used primarily for debugging, and so defaults to FALSE.

Details

Each element of the returned list is a metaprep object summarizing effect size data for a single gene. This list of metaprep gene information is passed to the metahdep() function for meta-analysis.

Value

A list, where each element is a gene-specific object of class `metaprep`.

Author(s)

John R. Stevens, Gabriel Nicholas

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
##  load a pre-made list of ES.obj's and newnames data.frame.  These objects hold
##  data in a format suitable for use in the metahdep.format function.
data(HGU.DifExp.list)
data(HGU.newnames)

##  now call the format function;
##  this may take anywhere from several seconds to several minutes,
##  depending on the speed of the computer and the number of genes under
##  consideration
HGU.prep.list <- metahdep.format(HGU.DifExp.list, HGU.newnames)
```

`metahdep.HBLM` *metahdep.HBLM*

Description

Performs a meta-analysis by fitting a hierarchical Bayes linear model, allowing for hierarchical dependence.

Usage

```
metahdep.HBLM(theta, V, X, M = NULL, dep.groups = NULL,
              meta.name = "meta-analysis", center.X = FALSE,
              delta.split = FALSE, n = 10, m = 10,
              two.sided = FALSE)
```

Arguments

`theta`	A vector of effect size estimates from multiple studies.
`V`	The variance/covariance matrix for `theta`. Typically, this will be block diagonal (to represent any sampling dependence).
`X`	A matrix of covariates for `theta`. At the very least, this must consist of an intercept term. Other covariates can be included, but there must be more rows than columns in this covariate matrix.

M	(optional) Used when delta.split=TRUE. A block-diagonal matrix describing the hierarchical dependence for the studies (theta). One of two ways to specify this is by using the metahdep.format() function; the other is to use the get.M() function.
dep.groups	(optional) Used when delta.split=TRUE. A list of vectors/scalars describing the hierarchical dependence groups for the studies (theta). This is an alternative to passing an M matrix.
meta.name	(optional) A name field for bookkeeping. This can be any character string.
center.X	(optional) A logical value specifying whether or not to center the columns of X. If TRUE, then the mean from each column will be subtracted from every element in that column (but not for the intercept). This changes the interpretation of the intercept coefficient estimate from the model fit.
delta.split	(optional) A logical value specifying whether or not to account for hierarchical dependence (i.e., perform delta-splitting). If TRUE, then the user needs to pass either a dependence matrix M, or a dep.groups list; i.e., one of M or dep.groups is REQUIRED when delta.split=TRUE.
n	(optional) An even integer telling how many steps to use when doing the numerical integration over tau, the square root of the between-study hierarchical variance. The integration is done on the log-logistic prior, split into the 4 quartiles. This number n specifies how many steps to take within each quartile.
m	(optional) An even integer telling how many steps to use when doing the numerical integration over varsigma (given tau), the between-study hierarchical covariance. This is only used when delta.split=TRUE. The integration is done on the uniform prior, for each value of tau.
two.sided	(optional) A logical value to determine whether to return the 2-sided p-values or default [one-sided positive] posterior probabilities for the parameter estimates.

Details

Takes a vector of effect size estimates, a variance/covariance matrix, and a covariate matrix, and fits a hierarchical Bayes linear model. If delta.split=TRUE, then it performs delta-splitting to account for hierarchical dependence among studies. The main parameters (beta) are given normal priors, the square root of the hierarchical variance (tau) is given a log-logistic prior, and the hierarchical covariance (varsigma) is given a uniform prior; see the Stevens and Taylor reference for details. When a meta-analysis is to be performed for gene expression data (on a per-gene basis), the metahdep() function calls this metahdep.HBLM function for each gene separately.

Value

A list, with the following named components:

beta.hats	A vector of model estimates for the covariates given by X (it may be length 1 i.e. scalar)
cov.matrix	The variance/covariance matrix for the beta.hats vector
beta.hat.p.values	The p-value(s) for the beta.hats estimate(s)
tau.hat	The posterior mean for tau (not tau-square). An estimate for tau-square is E(square(tau) [given data]) = tau.var + square(tau.hat)
tau.var	The posterior variance for tau (not tau-square).
varsigma.hat	The posterior mean for varsigma.

`varsigma.var` The posterior variance for varsigma.

`tau.varsigma.cov` The posterior covariance for tau and varsigma.

`name` An optional name field

Author(s)

John R. Stevens, Gabriel Nicholas

References

DuMouchel W. H. and Harris J. H. (1983), Bayes methods for combining the results of cancer studies in humans and other species, *Journal of the American Statistical Association*, 78(382), 293-308.

DuMouchel W.H. and Normand S.-L. (2000), Computer-modeling and graphical strategies for meta-analysis, in D. K. Stangl and D. A. Berry (Eds.), *Meta-analysis in medicine and health policy*, pp. 127-178. New York: Marcel Dekker.

Stevens J.R. and Doerge R.W. (2005), A Bayesian and Covariate Approach to Combine Results from Multiple Microarray Studies, *Proceedings of Conference on Applied Statistics in Agriculture*, pp. 133-147.

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

Stevens J.R. and Taylor A.M. (2009), Hierarchical Dependence in Meta-Analysis, *Journal of Educational and Behavioral Statistics*, 34(1):46-73.

See also the *metahdep* package vignette.

Examples

```
###
### Example 1: gene expression data
### - this uses one gene from the HGU.prep.list object

# load data and extract components for meta-analysis (for one gene)
data(HGU.prep.list)
gene.data <- HGU.prep.list[[7]]
theta <- gene.data@theta
V <- gene.data@V
X <- gene.data@X
M <- gene.data@M
dep.grps <- list(c(1:2),c(4:6))
gene.name <- gene.data@gene

# fit a regular HBLM (no hierarchical dependence)
results <- metahdep.HBLM(theta, V, X, meta.name=gene.name,
  center.X=TRUE, two.sided=TRUE)
results

# fit hierarchical dependence model (with delta-splitting),
# using two different methods for specifying the dependence structure
results.dsplitM <- metahdep.HBLM(theta, V, X, M, delta.split=TRUE,
  meta.name=gene.name, center.X=TRUE, two.sided=TRUE)
results.dsplitM
results.dsplitd <- metahdep.HBLM(theta, V, X, dep.groups=dep.grps,
```

```
  delta.split=TRUE, meta.name=gene.name, center.X=TRUE, two.sided=TRUE)
results.dsplitd


###
###  Example 2: glossing data
###  - this produces part of Table 5 in the Stevens and Taylor JEBS paper.

data(gloss)
dep.groups <- list(c(2,3,4,5),c(10,11,12))
HBLM.ds <- metahdep.HBLM(gloss.theta, gloss.V, gloss.X, center.X=TRUE,
  two.sided=TRUE, delta.split=TRUE, dep.groups=dep.groups, n=20, m=20)
round(cbind(HBLM.ds$beta.hats, HBLM.ds$beta.hat.p.values),4)
```

`metahdep.other` *metahdep.other*

Description

Miscellaneous functions used internally by the *metahdep* package's main functions (`metahdep`, `metahdep.FEMA`, `metahdep.REMA`, `metahdep.HBLM`, and `metahdep.format`):

`metahdep.list2dataframe`	convert list to `data.frame`
`LinMod.MetAn.dep.REMA`	REMA meta-analysis
`LinMod.REMA.dep`	used by `LinMod.MetAn.dep.REMA` to estimate parameters
`LinMod.REMA.delta.split`	REMA (with delta-splitting)
`LinMod.HBLM.fast.dep`	HBLM (no delta-splitting)
`new.LinMod.HBLM.fast.dep.delta.split`	HBLM (with delta-splitting)
`LinMod.MetAn.dep.FEMA`	FEMA
`metahdep.check.X`	check design matrix X, and drop columns if necessary to make full rank
`get.M`	create block diagonal M matrix, given dependence structure
`tr`	calculate trace of matrix
`id`	create identity matrix
`center.columns`	center all non-intercept columns of design matrix X
`mod`	mod function
`get.varsigma.v`	get varsigma values for HBLM delta-splitting model

Author(s)

John R. Stevens, Gabriel Nicholas

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

Stevens J.R. and Taylor A.M. (2009), Hierarchical Dependence in Meta-Analysis, *Journal of Educational and Behavioral Statistics*, 34(1):46-73.

See also the *metahdep* package vignette.

Examples

```
## Create the M matrix for the glossing example
## - here, studies 2-5 are one hierarchically dependent group (Baumann),
## and studies 10-12 are another hierarchically dependent group (Joyce)
data(gloss)
dep.groups <- list(c(2:5),c(10:12))
M <- get.M(length(gloss.theta),dep.groups)
```

`metahdep.REMA` *metahdep.REMA*

Description

Performs a random effects linear model meta-analysis, allowing for hierarchical dependence. It returns a list containing the results.

Usage

```
metahdep.REMA(theta, V, X, M = NULL, dep.groups = NULL,
              meta.name = "meta-analysis", delta.split = FALSE,
              center.X = FALSE)
```

Arguments

`theta`	A vector of effect size estimates from multiple studies.
`V`	The variance/covariance matrix for `theta`. Typically, this will be block diagonal (to represent any sampling dependence).
`X`	A matrix of covariates for `theta`. At the very least, this must consist of an intercept term. Other covariates can be included, but there must be more rows than columns in this covariate matrix.
`M`	(optional) Used when `delta.split=TRUE`. A block-diagonal matrix describing the hierarchical dependence for the studies (`theta`). One of two ways to specify this is by using the `metahdep.format()` function; the other is to use the `get.M()` function.
`dep.groups`	(optional) Used when `delta.split=TRUE`. A list of vectors/scalars describing the hierarchical dependence groups for the studies (`theta`). This is an alternative to passing an `M` matrix.
`meta.name`	(optional) A name field for bookkeeping. This can be any character string.
`delta.split`	(optional) A logical value specifying whether or not to account for hierarchical dependence (i.e., perform delta-splitting). If `TRUE`, then the user needs to pass either a dependence matrix `M`, or a `dep.groups` list; i.e., one of `M` or `dep.groups` is REQUIRED when `delta.split=TRUE`.
`center.X`	(optional) A logical value specifying whether or not to center the columns of `X`. If `TRUE`, then the mean from each column will be subtracted from every element in that column (but not for the intercept). This changes the interpretation of the intercept coefficient estimate from the model fit.

Details

Takes a vector of effect size estimates, a variance/covariance matrix, and a covariate matrix, and fits a random effects linear model meta-analysis, allowing for hierarchical dependence. If `delta.split=TRUE`, then it performs delta-splitting to account for hierarchical dependence among studies. When a meta-analysis is to be performed for gene expression data (on a per-gene basis), the `metahdep()` function calls this `metahdep.REMA()` function for each gene separately.

Value

A list, with the following named components:

`beta.hats`	A vector of model estimates for the covariates given by `X` (it may be a scalar, i.e., a vector of length 1)
`cov.matrix`	The variance/covariance matrix for the `beta.hats` vector
`beta.hat.p.values`	The [two-sided] p-value(s) for the `beta.hats` estimate(s)
`tau2.hat`	The estimated between-study hierarchical variance tau-square, using the method of moments approach of DerSimonian and Laird.
`varsigma.hat`	(Only estimated when `delta.split=TRUE`.) The estimated within-group hierarchical covariance.
`Q`	The statistic used to test for model homogeneity / model mis-specification
`Q.p.value`	The p-value for `Q`
`name`	An optional name field

Author(s)

John R. Stevens, Gabriel Nicholas

References

DerSimonian R. and Laird N. (1986), Meta-analysis in clinical trials, *Controlled Clinical Trials*, 7: 177-188.

Hedges L. V. and Olkin I (1985), *Statistical methods for meta-analysis*, San Diego, CA: Academic Press.

Stevens J.R. and Doerge R.W. (2005), Combining Affymetrix Microarray Results, *BMC Bioinformatics*, 6:57.

Stevens J.R. and Taylor A.M. (2009), Hierarchical Dependence in Meta-Analysis, *Journal of Educational and Behavioral Statistics*, 34(1):46-73.

See also the *metahdep* package vignette.

Examples

```
###
### Example 1: gene expression data
### - this uses one gene from the HGU.prep.list object

# load data and extract components for meta-analysis (for one gene)
data(HGU.prep.list)
gene.data <- HGU.prep.list[[7]]
theta <- gene.data@theta
V <- gene.data@V
```

```
X <- gene.data@X
M <- gene.data@M
dep.grps <- list(c(1:2),c(4:6))
gene.name <- gene.data@gene

# fit a regular REMA (no hierarchical dependence)
results <- metahdep.REMA(theta, V, X, meta.name=gene.name)
results

# fit hierarchical dependence model (with delta-splitting),
# using two different methods for specifying the dependence structure
results.dsplitM <- metahdep.REMA(theta, V, X, M, delta.split=TRUE,
  meta.name=gene.name, center.X=TRUE)
results.dsplitM
results.dsplitd <- metahdep.REMA(theta, V, X, dep.groups=dep.grps,
  delta.split=TRUE, meta.name=gene.name, center.X=TRUE)
results.dsplitd


###
###  Example 2: glossing data
###  - this produces part of Table 6 in the Stevens and Taylor JEBS paper.

data(gloss)
dep.groups <- list(c(2,3,4,5),c(10,11,12))
REMA.ds <- metahdep.REMA(gloss.theta, gloss.V, gloss.X, center.X=TRUE,
  delta.split=TRUE, dep.groups=dep.groups)
round(cbind(t(REMA.ds$beta.hats), sqrt(diag(REMA.ds$cov.matrix)),
  t(REMA.ds$beta.hat.p.values)),4)
```

`metaprep-class` *Class metaprep*

Description

This is a class representation for the effect size estimates and other summary information for a single gene, from a collection of gene expression studies, usually constructed in preparation for a meta-analysis.

Objects from the Class

Objects can be created using the function `metahdep.format` and `new`.

Slots

`theta`**:** Object of class `vector` representing the gene's effect size estimates (differential expression measures) from the multiple studies.

`V`**:** Object of class `matrix` representing the sampling variance/covariance matrix of the gene's effect size estimates from the multiple studies.

`X`**:** Object of class `matrix` representing the covariate (or design) matrix for the gene. Covariate information from the multiple studies is represented here.

`M`: Object of class `matrix` representing the block diagonal hierarchical structure of effect size estimates from the multiple studies.

`max.k`: Object of class `integer` representing the size of the largest block on the diagonal of `M`, i.e., the size of the largest hierarchical dependence group for the gene.

`row.indices`: Object of class `matrix` with columns named Study and Row. Optionally returned by the function `metahdep.format()`, to see which gene (Row) in which `ES.obj` object (Study) produced the data recorded in each `metaprep` object.

`gene`: Object of class `character` representing the gene name.

Methods

@ replace the slot entries

References

Stevens J.R. and Nicholas G. (2009), metahdep: Meta-analysis of hierarchically dependent gene expression studies, *Bioinformatics*, 25(19):2619-2620.

See also the *metahdep* package vignette.

Examples

```
###
###  See the metahdep package vignette for a full example
###
data(HGU.prep.list)
HGU.prep.list[[7]]
```

Index

∗ **classes**
  ES.obj-class, 2
  metaprep-class, 19
∗ **datasets**
  gloss, 4
  HGU.DifExp.list, 5
  HGU.newnames, 6
  HGU.prep.list, 7
∗ **htest**
  getPLM.es, 3
  metahdep, 8
  metahdep.FEMA, 10
  metahdep.HBLM, 13
  metahdep.REMA, 17
∗ **models**
  getPLM.es, 3
  metahdep, 8
  metahdep.FEMA, 10
  metahdep.format, 12
  metahdep.HBLM, 13
  metahdep.other, 16
  metahdep.REMA, 17

center.columns *(*metahdep.other*)*, 16

ES.obj-class, 2

get.M *(*metahdep.other*)*, 16
get.varsigma.v *(*metahdep.other*)*, 16
getPLM.es, 3
gloss, 4

HGU.DifExp.list, 5
HGU.newnames, 6
HGU.prep.list, 7

id *(*metahdep.other*)*, 16

LinMod.HBLM.fast.dep *(*metahdep.other*)*, 16
LinMod.MetAn.dep.FEMA *(*metahdep.other*)*, 16
LinMod.MetAn.dep.REMA *(*metahdep.other*)*, 16
LinMod.REMA.delta.split *(*metahdep.other*)*, 16
LinMod.REMA.dep *(*metahdep.other*)*, 16

metahdep, 8
metahdep.check.X *(*metahdep.other*)*, 16
metahdep.FEMA, 10
metahdep.format, 12
metahdep.HBLM, 13
metahdep.list2dataframe *(*metahdep.other*)*, 16
metahdep.other, 16
metahdep.REMA, 17
metaprep-class, 19
mod *(*metahdep.other*)*, 16

new.LinMod.HBLM.fast.dep.delta.split *(*metahdep.other*)*, 16

tr *(*metahdep.other*)*, 16