Package 'mixmeta'

May 8, 2026

Type Package

Version 1.2.2

Date 2025-10-07

Title An Extended Mixed-Effects Framework for Meta-Analysis

Description A collection of functions to perform various meta-analytical models through a unified mixed-effects framework, including standard univariate fixed and random-effects meta-analysis and meta-regression, and non-standard extensions such as multivariate, multilevel, longitudinal, and dose-response models.

Maintainer Antonio Gasparrini <antonio.gasparrini@lshtm.ac.uk>

Imports stats, graphics, grDevices, utils

Depends R (>= 3.5.0)

Suggests metafor, dosresmeta, nlme, MASS, dlnm

URL https://github.com/gasparrini/mixmeta, http://www.ag-myresearch.com/package-mixmeta

License GPL (>= 3)

LazyData yes

Encoding UTF-8

NeedsCompilation no

Author Antonio Gasparrini [cre, aut, cph] (ORCID: <https://orcid.org/0000-0002-2271-3568>), Francesco Sera [aut]

Repository CRAN

Date/Publication 2025-10-13 08:50:02 UTC

Contents

mixmeta-package *An Extended Mixed-Effects Framework for Meta-Analysis*

Description

The package **mixmeta** consists of a collection of functions to perform various meta-analytical models in R through a unified mixed-effects framework, including standard univariate fixed and random-effects meta-analysis and meta-regression, and non-standard extensions such as multivariate, multilevel, longitudinal, and dose-response models.

Modelling framework

Standard applications of meta-analysis amount to the pooling of estimates of a single effect size, here defined generally as outcome, collected as unique observations in a set of independent studies, together with a measure of uncertainty (usually standard errors). Fixed-effects models do not assume heterogeneity across studies, and the estimates are conditional on the set of studies collected in the meta-analysis. Random-effects meta-analysis, instead, allows a degree of heterogeneity among studies, assuming the (true but unobserved) study-specific outcomes as randomly sampled from a (usually hypothetical) population of studies. Meta-regression extends both fixed and random-effects methods by allowing the pooled outcome to depend on study-level meta-predictors.

However, this traditional setting can be limited for many modern applications of meta-analysis. For instance, studies can provide estimates of different outcomes. Alternatively, studies can collect multiple estimates of the same outcome, either longitudinally or referring to different groups or levels of a continuous variable. Similarly, studies can be clustered, or being characterized by a hierarchical structure (i.e., by country). In all these instances, the key assumption of independence across estimates is not met, and basic models must be extended to consider potentially complex correlation structures within and between studies. This leads to extension to multivariate, multilevel, longitudinal, or dose-response models for meta-analysis, among others.

A unified modelling framework can be defined by casting the meta-analytical problem as a linear mixed model. In general terms, we assume that there is a set of n observations of k different outcomes, representing *units* of analysis aggregated in $i = 1, \ldots, m$ *groups* that are considered independent. An extended random-effects meta-regression model for the $\boldsymbol{y}_i$ outcomes in group i can be generally written as:

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i\boldsymbol{b} + \boldsymbol{\epsilon}_i$$
$$\boldsymbol{b} \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{\Psi}), \boldsymbol{\epsilon}_i \sim \mathrm{N}(\boldsymbol{0}, \boldsymbol{S}_i)$$

Here, $\boldsymbol{X}_i\boldsymbol{\beta}$ defines the fixed effects that represent the population-averaged outcomes in terms of p unit-level predictors in the design matrix $\boldsymbol{X}_i$ with fixed-effects coefficients $\boldsymbol{\beta}$. The random part of the model $\boldsymbol{Z}_i\boldsymbol{b}$ describes the deviation from the population averages in terms of q unit-level predictors in the design matrix $\boldsymbol{Z}_i$ and random-effects coefficients $\boldsymbol{b}$. The marginal (co)variance matrix $\boldsymbol{\Sigma}_i = \boldsymbol{Z}\boldsymbol{\Psi}\boldsymbol{Z}^t + \boldsymbol{S}_i$ is given by the sum of within (assumed known) and between-group contributions, defined by (co)variance matrices $\boldsymbol{S}_i$ and $\boldsymbol{\Psi}$, respectively.

All the models mentioned above, and other extensions, can be described as special cases of this unified framework. Specifically, in the standard random-effects univariate meta-analysis or meta-regression, each group represents a study with a single observation ($n = m$), where $\boldsymbol{Z_i} = \boldsymbol{1}$

($q = 1$), and $\boldsymbol{y}_i$, $\boldsymbol{S}_i$ and $\boldsymbol{\Psi}$ are scalars. In fixed-effects models, $\boldsymbol{\Psi}$ and $\boldsymbol{Z_i}$ do not exist. In multivariate models, the k-dimensional $\boldsymbol{y}_i$ represents the different outcomes from study i, $\boldsymbol{X}_i$ is Kronecker-expanded to $k \times kp$, and $\boldsymbol{S}_i$ and $\boldsymbol{\Psi}$ are $k \times k$ matrices representing within and between-study correlations among outcomes, respectively. In multilevel models, where additional inner levels of grouping exist within each of the m outer-level groups, q is the sum of level-specific meta-predictors, while $\boldsymbol{\Psi}$ and $\boldsymbol{Z}_i$ have a block-diagonal and column-binded (and expanded) forms, respectively, with each part referring to a different level. In longitudinal and dose-response models, repeated measures are accomodated in a similar way through random-effects grouping.

Estimation methods

The aim is to estimate the kp coefficients $\boldsymbol{\beta}$ and, for random-effects models, the components of the between-group (co)variance matrix $\boldsymbol{\Psi}$. The parameters for the random part depend on the number of random-effects levels, and for each of them, on the number of random-effects meta-predictors and the structure of the related part of the (co)variance matrix, with a maximum of $kq(kq+1)/2$ for single-level unstructured $\boldsymbol{\Psi}$.

Different estimators are implemented in the package **mixmeta**. The options available in the current version are:

- **Fixed-effects**
- **Maximum likelihood** (ML)
- **Restricted maximum likelihood** (REML)
- **Method of moments**
- **Variance components**

The fixed-effects model is fitted through generalized least squares (GLS), assuming the (co)variance structure, composed by the within-study errors only, as completely known. Likelihood-based random-effects estimators, ML and REML, represent the most comprehensive implementation of the modelling framework, and allow the specification all the various models described in the previous section through a flexible definition of the random-effects structure. They rely on two alternative iterative optimization procedures, based on Newton-type and (restricted) iterative generalized least squares (IGLS and RIGLS) algorithms, respectively. Estimators based on semiparametric alternatives such as the non-iterative method of moments or the iterative variance components are also included, although they are only available for models with a basic random-effects structure. Further details on estimation methods are given in the related help pages.

Functions included in the package

The main function in the package is `mixmeta`, which performs the various models illustrated above. This function resembles standard regression functions in R, and specifies the model through regression formulae for fixed and random-effects (see `mixmetaFormula`). The function returns a list object of class `"mixmeta"` (see `mixmetaObject`).

The estimation is carried out internally through `mixmeta.fit`, a wrapper which accepts data in a specific format, then prepares the various data components and calls ad hoc estimation functions for fitting the models. Specifically, `mixmeta.fixed` is applied for fixed-effects models, while estimators for random-effects models are implemented in the functions `mixmeta.ml` and `mixmeta.reml` for (restricted) maximum likelihood, `mixmeta.mm` for the method of moments, and `mixmeta.vc`

for variance components. For likelihood-based methods, alternative iterative optimizations methods are provided in two sets of functions implementing Newton-type and (R)IGLS algorithms used for maximizing the (restricted) likelihood. The former method applies specific likelihood functions. Various types of likelihood-based models are defined by separate regression formulae for fixed and random-effects (see mixmetaFormula). Specific (co)variance structures for the between-group random effects at single or multiple levels are available. Fitting parameter options are set by mixmeta.control.

Method functions are available for objects of class "mixmeta" (see mixmetaObject for a complete list). The method summary produces a list of class "summary.mixmeta" for summarizing the fit of the model and providing additional results. The method function predict computes predicted values, optionally for a set of new values of the predictors. blup gives the (empirical) best linear unbiased predictions for the set of studies used for estimation. Other default or specific method functions for regression can be used on objects of class "mixmeta", such as fitted and residuals, logLik, AIC and BIC, or drop1 and add1, among others.

Methods for model.frame, model.matrix, and terms are used to extract or construct the model frame, the design matrix, or the terms of the regression meta-analytical model, respectively. These specific methods for objects of class "mixmeta" are needed to appropriately deal with missing values and to account for model frames that include terms for both the fixed and random parts. In particular, methods for na.omit and na.exclude are used to handle correctly missing values.

Simulations can be produced using the function mixmetaSim and the method function simulate, which return one or multiple sets of simulated outcomes for a group of studies. The function inputna and inputcov are used internally to augment the missing data values and to input missing correlations, respectively.

The method function qtest.mixmeta (producing an object with class of the same name) performs the (multivariate) Cochran Q test for (residual) heterogeneity. For multivariate models, the function returns both an overall estimate and those for each single outcome. The generic method function is qtest.

Printing functions for the objects of classes defined above are also provided. Other functions are used internally in the source code, and not exported in the namespace. For users interested in getting into details of the package structure, these functions can be displayed using the triple colon (':::') operator. For instance, mixmeta:::glsfit displays the code of the function glsfit. Also, some comments are added in the original source code.

Datasets and applications

The package includes several datasets used for applications of the extended meta-analytical framework. The related help pages provide examples of specific models, and fully demonstrate the flexibility of the extended meta-analytical framework. In particular:

- **Standard meta-analysis** is illustrated using the dataset bcg, including examples of meta-regression.
- **Multivariate meta-analysis** is performed using various datasets, including bivariate models (berkey98, hyp, p53) and multivariate models with three or more outcomes (fibrinogen and hsls). The examples describe also how to deal with missing data or missing within-group correlations, and multivariate meta-regression.
- **Network meta-analysis** is shown in the dataset smoking. The examples illustrate an indirect mixed-treatment comparison including consistency and inconsistency models.

- **Multilevel meta-analysis** is displayed in the examples of the datasets `school` and `thrombolytic`, and include data with multiple nested levels of grouping and/or repeated measures within each group.
- **Dose-response meta-analysis** is illustrated in the dataset `alcohol`, using the recently proposed one-stage approach.
- **Longitudinal meta-analysis** is performed using the datasets `dbs` and `gliomas`. The two sets of examples present different cases using data in wide and long format, respectively.

Additional information

The **mixmeta** package is available on the Comprehensive R Archive Network (CRAN), with info at the related web page (CRAN.R-project.org/package=mixmeta). A development website is available on GitHub (github.com/gasparrini/mixmeta). General information on the development and applications of this extended meta-analytical modelling framework, together with an updated version of the R scripts for running the examples in published papers, can be found in GitHub (github.com/gasparrini) or at the personal web page of the package maintainer (www.ag-myresearch.com).

The package **mixmeta** is an extension of the package **mvmeta**, previously developed to perform multivariate meta-analytical models. The latter now depends on the former, and while both are still maintained, users are encouraged to switch to **mixmeta** as it represents a more general and updated option. A list of changes included in the current and previous versions of **mixmeta** can be found by typing:

```
news(package="mixmeta")
```

Use `citation("mixmeta")` to cite this package.

Warnings

This release of the package **mixmeta** has been tested with different simulated and real datasets. The functions generally perform well under several scenarios, and comparisons with alternative software implementations show good agreement. However, bugs and bad performance under untested conditions may not be excluded. Please report any error or unexpected behaviour to the e-mail address below.

Note

The package **mixmeta** provides a unified modelling framework to perform standard and non-standard meta-analytical models. However, some of these can also be fitted using routines available in other R packages.

For instance, many packages such as **metafor**, **meta**, **rmeta** provide a more exhaustive and efficient set of methods for standard univariate meta-analysis and meta-regression, including a wide range or functions for specific plots and statistical tests.

Specific modelling extensions are also provided by other packages. For example, multivariate or multilevel models can be also be fitted using functions in **metafor** and **metaSEM**, while dose-response meta-analysis and meta-analysis of diagnostic measures can be performed using **dosresmeta** and **mada**, respectively.

See the CRAN Task View Meta-Analysis for a comprehensive illustration of methods available in various R packages.

Author(s)

Antonio Gasparrini and Francesco Sera

Maintainer: Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

Pinheiro JC and Bates DM (2000). *Mixed-Effects Models in S and S-PLUS*. New York, Springer Verlag.

Lindstrom MJ and Bates DM (1988). Newton-Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*. **83**(404):1014–1022.

Goldstein H (1986). Multilevel mixed linear model analysis using iterative generalized least squares. *Biometrika*. **73**(1):43–56.

Goldstein H (1992). Efficient computational procedures for the estimation of parameters in multilevel models based on iterative generalized least squares. *Computational Statistics & Data Analysis*. **13**(1):63–71.

Stram DO (1996). Meta-analysis of published data using a linear mixed-effects model. *Biometrics*. **52**(2):536–544.

Stevens JR, Taylor AM. Hierarchical dependence in meta-analysis. *Journal of Educational and Behavioral Statistics*. **34**(1):46–73.

Jackson D, Riley R, White IR (2011). Multivariate meta-analysis: Potential and promise. *Statistics in Medicine*. **30**(20);2481–2498.

Goldstein H, et al (2000). Meta-analysis using multilevel models with an application to the study of class size effects. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*. **49**(3):399–412.

Crippa A, et al (2019). One-stage dose-response meta-analysis for aggregated data. *Statistical Methods in Medical Research*. **28**(5):1579–1596.

Ishak KJ, Platt RW, Joseph L, et al (2007). Meta-analysis of longitudinal studies. *Clinical Trials*. **4**(5):525–539.

`alcohol` *Alcohol Intake and Colorectal Cancer*

Description

The dataset contains the data on 8 cohort studies participating in the Pooling Project of Prospective Studies of Diet and Cancer. A total of 3,646 cases and 2,511,424 person-years were included in the analysis. Each study estimated the incidence relative rate in different categories of alcohol intake while controlling for a set of potential confounders, using non-drinkers as the reference. The categories where then converted in a dose by assigning to each the median value of individual consumptions, with studies reporting estimates at different levels in a continuous scale.

Usage

```
alcohol
```

Format

A data frame with 48 observations on the following 7 variables:

- `id`: label for each study, derived from the first author's name.
- `type`: code for study design (cohort estimating incidence rate).
- `dose`: assigned dose level (gr/day of alcohol intake).
- `cases`: number of cases for each dose category.
- `peryears`: amount of person-time for each dose category.
- `logrr`: estimated logarithm of the incidence relative rate.
- `se`: standard error of the estimates.

Details

The data are stored in a *long* format, with each record reporting the information for each dose categories and studies including multiple records. The reference category for each study included, although the log-RR is fixed to 0 with no standard error (comparing the category with itself). The information on these reference categories is needed to compute the approximate correlations between estimates in the same study.

Note

The data provide an example of application of dose-response meta-analysis, with repeated measurements of the effect size associated to different doses within each study. This requires a modelling structure that accounts for both within and between-study correlations of repeated measurements. The within-study correlations are usually reconstructed from published data using specific methods. Results can be compared with those reported by Crippa and Orsini (2016) and Orsini and colleagues (2012), although they are not identical: while the original analysis used a two-stage approach, the modelling framework applied here follows the more recent one-stage dose-response meta-analysis proposed by Crippa and collegues (2019).

The dataset is also available in the same format in the dataframe `alcohol_crc` of the package **dosresmeta**.

Source

Crippa A, et al (2019). One-stage dose-response meta-analysis for aggregated data. *Statistical Methods in Medical Research*. **28**(5):1579–1596.

Crippa A, Orsini N (2016). Multivariate dose-response meta-analysis: The dosresmeta R package. *Journal of Statistical Software*. **72**(1):1–15.

Orsini N, et al (2012). Meta-analysis for linear and nonlinear dose-response relations: examples, an evaluation of approximations, and software. *American Journal of Epidemiology*. **175**(1):66–73.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN CRIPPA ET AL (2016) AND ORSINI ET AL (2012)

# LOAD THE PACKAGE dosresmeta AND splines
library(dosresmeta) ; library(splines)

# COMPUTE THE WITHIN-STUDY CORRELATIONS EXCLUDING THE REFERENCE
addS <- lapply(split(alcohol, alcohol$id), function(x)
  covar.logrr(y=logrr, v=se^2, cases=cases, n=peryears, type=type, data=x))
sub <- subset(alcohol, !is.na(se))

# NOT ACCOUNTING FOR WITHIN-STUDY CORRELATIONS
nocor <- mixmeta(logrr ~ 0 + dose, S=se^2, random= ~ 0 + dose|id, data=sub,
  method="ml")
summary(nocor)

# ACCOUNTING FOR WITHIN-STUDY CORRELATIONS
lin <- mixmeta(logrr ~ 0 + dose, random= ~ 0 + dose|id, data=sub, method="ml",
  control=list(addSlist=addS))
summary(lin)

# ALLOWING NON-LINEARITY IN BOTH FIXED AND RANDOM PARTS
nonlin <- mixmeta(logrr ~ 0 + ns(dose, knots=c(10,25)), data=sub,
  random= ~ 0 + ns(dose, knots=c(10,25))|id, method="ml",
  control=list(addSlist=addS))
summary(nonlin)

# SIMPLIFY THE MODEL BY ASSUMING LINEARITY IN THE RANDOM PART
nonlin2 <- update(nonlin, random= ~ 0 + dose|id)
summary(nonlin2)

# FIXED-EFFECTS MODEL (TRICK: random TO DEFINE THE GROUPING, THEN FIX IT TO 0)
nonlinfix <- mixmeta(logrr ~ 0 + ns(dose, knots=c(10,25)), random= ~ 1|id,
  data=sub, method="ml",bscov="fixed", control=list(addSlist=addS, Psifix=0))
summary(nonlinfix)

# COMPARE THE MODELS
AIC(nocor, lin, nonlin, nonlin2, nonlinfix)
```

```
# PREDICT THE RR FOR 12g/day FOM TWO MODELS
exp(predict(nocor, newdata=data.frame(dose=12), ci=TRUE))
exp(predict(lin, newdata=data.frame(dose=12), ci=TRUE))

# PREDICT (RECREATE SPLINES FOR EASY CODING)
predlin <- exp(predict(lin, newdata=data.frame(dose=0:60), ci=TRUE))
prednonlin <- exp(predict(nonlin, newdata=data.frame(dose=0:60), ci=TRUE))

# DISPLAY THE NON-LINEAR EFFECT
col1 <- do.call(rgb, c(as.list(col2rgb("blue") / 255), list(0.2)))
col2 <- do.call(rgb, c(as.list(col2rgb("green") / 255), list(0.2)))
plot(0:60, predlin[,1], type="l", ylim=c(0.85,1.9), ylab="RR",
  xlab="Alcohol intake (gr/day)", main="Dose-response")
polygon(c(0:60,60:0), c(predlin[,2], rev(predlin[,3])), col=col1, border=NA)
lines(0:60,prednonlin[,1], lty=5)
polygon(c(0:60,60:0), c(prednonlin[,2],rev(prednonlin[,3])), col=col2, border=NA)
```

---

bcg *Efficacy of BCG Vaccine in the Prevention of Tuberculosis*

Description

The dataset contains the data on 13 prospective clinical trials that compared the rates of tubercolosis in groups vaccinated with the Bacillus Calmette-Guerin (BCG) vaccine and non-vaccinated control populations. The outcome here is reported as both relative risk (RR) and odds ratio (OR), with associated uncertanty.

Usage

```
bcg
```

Format

A data frame with 13 observations on the following 13 variables:

- `trial`: sequence identifying the trial.
- `author`: label identifying the author(s).
- `year`: year of publication.
- `tpos, tneg`: number of positive and negative TB cases in the treated (vaccinated) group.
- `cpos, cneg`: number of positive and negative TB cases in the control (non-vaccinated) group.
- `ablat`: absolute latitude of the study location (in degrees).
- `alloc`: method of treatment allocation (random, alternate, or systematic assignment).

Note

The data provide an example of application of standard univariate meta-analysis and meta-regression, with independent studies providing a single estimate of a single effect size. Interestingly, the data can be analyzed also as a multivariate meta-analysis, using a bivariate outcome where risks or odds of TB can be measured separaterly in treatment and control groups. Results can be compared with those reported van Houwelingen, Arends, and Stijnen (2002).

The dataset is also available in the same format in the dataframe `dat.colditz1994` of the package **metafor**.

Source

van Houwelingen HC, Arends LR, Stijnen T (2002). Advanced methods in meta-analysis: multivariate approach and meta-regression. *Statistics in Medicine*. **21**(4):589–624.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN VAN HOUWELINGEN ET AL (2002)

# FIXED-EFFECTS META-ANALYSIS (SECTION 3.1.1)
unifix <- mixmeta(logor, logorvar, data=bcg, method="fixed")
print(summary(unifix), digits=3)

# RANDOM-EFFECTS META-ANALYSIS WITH MAXIMUM LIKELIHOOD (SECTION 3.1.2)
uniran <- mixmeta(logor, logorvar, data=bcg, method="ml")
print(summary(uniran), digits=3, report="var")

# ORIGINAL ESTIMATES AND BEST-LINEAR UNBIASED PREDICTIONS (FIGURE 3)
pred <- with(bcg, cbind(logor, logor-1.96*sqrt(logorvar),
  logor+1.96*sqrt(logorvar)))
blup <- blup(uniran, pi=TRUE)
plot(pred[,1], rev(bcg$trial)+0.2, xlim=c(-3,3), ylim=c(0,14), pch=18,
  axes=FALSE, xlab="Log odds ratio", ylab="Trial", main="Forest plot")
axis(1)
axis(2, at=bcg$trial, labels=rev(bcg$trial), lty=0, las=1)
abline(v=coef(uniran))
segments(pred[,2], rev(bcg$trial)+0.2, pred[,3], rev(bcg$trial)+0.2, lty=5)
points(blup[,1], rev(bcg$trial)-0.2, pch=19)
segments(blup[,2], rev(bcg$trial)-0.2, blup[,3], rev(bcg$trial)-0.2)

# COMPUTE THE OUTCOME SEPARATELY FOR TREATMENT AND CONTROL GROUPS
y <- with(bcg, log(cbind(tpos/tneg, cpos/cneg)))
S <- with(bcg, cbind(1/tpos+1/tneg, 1/cpos+1/cneg))

# BIVARIATE RANDOM-EFFECTS META-ANALYSIS (SECTION 4)
mvran <- mixmeta(y, S, method="ml")
print(summary(mvran), digits=3, report="var")

# META-REGRESSION (SECTION 5)
```

```
uniranlat <- update(uniran, .~. + ablat)
print(summary(uniranlat), digits=3, report="var")
drop1(uniranlat, test="Chisq")
```

---

bdiagMat *Block-Diagonal Expansion of a List of Matrices*

Description

The function bdiagMat builds a single matrix with block-diagonal from a list of matrices.

Usage

```
bdiagMat(x)
```

Arguments

x	a list of matrices, or a single matrix.

Value

A matrix with block-diagonal form if x is a list, or otherwise x itself if a matrix.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See functions bldiag in package **metafor**.

Examples

```
# GENERATE A LIST OF MATRICES, AND CREATE THE BLOCK-DIAGONAL MATRIX
(matlist <- list(matrix(1:4,2), matrix(1:8,2)))
bdiagMat(matlist)
```

berkey98 *Five Published Trials on Periodontal Disease*

Description

The dataset contains the results of 5 published trials comparing surgical and non-surgical treatments for medium-severity periodontal disease, one year after treatment. The 2 estimated outcomes are average improvements (surgical minus non-surgical, in mm) in probing depth (PD) and attachment level (AL).

Usage

```
berkey98
```

Format

A data frame with 5 observations on the following 7 variables:

`pubyear` publication year of the trial.

`npat` number of patients included in the trial.

`PD` estimated improvement of surgical versus non-surgical treatments in probing depth (mm).

`AL` estimated improvement of surgical versus non-surgical treatments in attachment level (mm).

`var_PD` variance of the estimated outcome for `PD`.

`cov_PD_AL` covariance of the estimated outcomes for `PD` and `AL`.

`var_AL` variance of the estimated outcome for `AL`.

Row names specify the author of the paper reporting the results of each trial.

Source

Berkey CS, et al. (1998). Meta-analysis of multiple outcomes by regression with random effects. *Statistics in Medicine*. **17**:2537–2550.

Berkey CS., et al. (1995). Multiple-outcomes meta-analysis of treatments for periodontal disease. *Journal of Dental Research*. **74**(4):1030–1039.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN BERKEY ET AL. (1998)

# INSPECT THE DATA
berkey98

# FIXED-EFFECTS
year <- berkey98$pubyear - 1983
```

```
mod1 <- mixmeta(cbind(PD,AL) ~ year, S=berkey98[5:7], data=berkey98,
  method="fixed")
print(summary(mod1), digits=3)

# GLS MODEL (VARIANCE COMPONENTS)
mod2 <- mixmeta(cbind(PD,AL) ~ year, S=berkey98[5:7], data=berkey98,
  method="vc",  control=list(vc.adj=FALSE))
print(summary(mod2), digits=3)
round(mod2$Psi, 3)

# ML MODEL
mod3 <- mixmeta(cbind(PD,AL) ~ year, S=berkey98[5:7], data=berkey98, method="ml")
print(summary(mod3), digits=3)
round(mod3$Psi, 3)
```

`blup` *Best Linear Unbiased Predictions*

Description

This is a generic function for generating best linear unbiased predictions (BLUPs) from the results of various fitting functions for meta-analytical models. The function invokes particular methods which depend on the `class` of the first argument. Currently, specific methods exist for several meta-analytical models in various packages: `blup.mixmeta`, `blup.rma.uni`, `blup.mvmeta`, and `blup.dosresmeta`.

Usage

```
blup(object, ...)
```

Arguments

`object`	a model object for which BLUPs are desired.
`...`	further arguments passed to or from other methods.

Details

The generic method function `blup` calls specific method functions to produces (empirical) best linear unbiased predictions (BLUPs) from model objects.

These predictions are a shrunk version of unit-specific realizations, where unit-specific estimates borrow strength from the assumption of an underlying (potentially multivariate) distribution in a (usually hypothetical) population. The amount of shrinkage depends from the relative size of the within and between-unit covariance matrices.

Value

The form of the value returned by `blup` depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method. Usually, the results consist of point estimates of BLUPs and optionally some measure of their uncertainty.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Verbeke G, Molenberghs G. *Linear Mixed Models for Longitudinal Data*. Springer; 1997.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

Specific methods for various classes: `blup.mixmeta`, `blup.rma.uni`, `blup.mvmeta`, and `blup.dosresmeta`.

`blup.mixmeta` *Best Linear Unbiased Predictions from mixmeta Models*

Description

This method function computes (empirical) best linear unbiased predictions from fitted random-effects meta-analytical models represented in objects of class `"mixmeta"`. Quantities can represent prediction of outcomes given both fixed and random effects, or just random-effects residuals from the fixed-effects estimates. Predictions are optionally accompanied by standard errors, prediction intervals or the entire (co)variance matrix of the predicted outcomes.

Usage

```
## S3 method for class 'mixmeta'
blup(object, se=FALSE, pi=FALSE, vcov=FALSE, pi.level=0.95, type="outcome",
  level, format, aggregate="stat", ...)
```

Arguments

`object`	an object of class `"mixmeta"`.
`se`	logical switch indicating if standard errors must be included.
`pi`	logical switch indicating if prediction intervals must be included.
`vcov`	logical switch indicating if the (co)variance matrix must be included.
`pi.level`	a numerical value between 0 and 1, specifying the confidence level for the computation of prediction intervals.
`type`	the type of prediction. This can be either `outcome` (default) or `residual`. See Details.
`level`	level of random-effects grouping for which predictions are to be computed. Default to the highest (inner) level, with 0 corresponding to fixed-effects predictions obtained through `predict`.
`format`	the format for the returned results. See Value.
`aggregate`	when `format="matrix"` and `se` or `ci` are required, the results may be aggregated by statistic or by outcome. See Value.
`...`	further arguments passed to or from other methods.

Details

The method function blup produces (empirical) best linear unbiased predictions from mixmeta objects. These can represent outcomes, given by the sum of fixed and random parts, or just random-effects residuals representing deviations from the fixed-effects estimated outcomes. In non-standard models with multiple hierarchies of random effects, the argument level can be used to determine the level of grouping for which predictions are to be computed.

These predictions are a shrunk version of unit-specific realizations, where unit-specific estimates borrow strength from the assumption of an underlying (potentially multivariate) distribution of outcomes or residuals in a (usually hypothetical) population. The amount of shrinkage depends from the relative size of the within and between-unit covariance matrices reported as components S and Psi in mixmeta objects (see mixmetaObject).

Fixed-effects models do not assume random effects, and the results of blup for these models are identical to predict (for type="oucome") or just 0's (for type="residuals").

How to handle predictions for units removed from estimation due to invalid missing pattern is determined by the na.action argument used in mixmeta to produce object. If na.action=na.omit, units excluded from estimation will not appear, whereas if na.action=na.exclude they will appear, with values set to NA for all the outcomes. This step is performed by napredict. See Note below.

In the presence of missing values in the outcomes y of the fitted model, correspondent values of point estimates and covariance terms are set to 0, while the variance terms are set to 1e+10. In this case, in practice, the unit-specific estimates do not provide any information (their weight is virtually 0), and the prediction tends to the value returned by predict with interval="prediction", when applied to a new but identical set of predictors. See also Note below.

Value

(Empirical) best linear unbiased predictions of outcomes or random-effects residuals. The results may be aggregated in matrices (the default), or returned as lists, depending on the argument format. For multivariate models, the aggregation is ruled by the argument aggregate, and the results may be grouped by statistic or by outcome. If vcov=TRUE, lists are always returned.

Note

The definition of missing in model frames used for estimation in mixmeta is different than that commonly adopted in other regression models such as lm or glm. See info on missing values in mixmeta.

Differently from predict, this method function computes the predicted values in the presence of partially missing outcomes. Interestingly, BLUPs for missing outcomes may be slightly different than predictions returned by predict on a new but identical set of predictors, as the BLUP also depends on the random part of the model. Specifically, the function uses information from the random-effects (co)variance to predict missing outcomes given the observed ones.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Verbeke G, Molenberghs G. *Linear Mixed Models for Longitudinal Data*. Springer; 1997.

See Also

See `predict` for standard predictions. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL) ~ 1, S=berkey98[5:7], data=berkey98)

# ONLY BLUP
blup(model)

# BLUP AND SE
blup(model, se=TRUE)

# SAME AS ABOVE, AGGREGATED BY OUTCOME, WITH PREDICTION INTERVALS
blup(model, se=TRUE, pi=TRUE, aggregate="outcome")

# WITH VCOV, FORCED TO A LIST
blup(model, se=TRUE, pi=TRUE, vcov=TRUE, aggregate="outcome")

# PREDICTING ONLY THE RANDOM-EFFECT RESIDUALS
blup(model, type="residual")
```

`coef.mixmeta` *Extract Coefficients and (Co)Variance Matrix from mixmeta Objects*

Description

These method functions return the estimated fixed-effects coefficients and their (co)variance matrix for fitted meta-analytical models represented in objects of class `"mixmeta"`.

Usage

```
## S3 method for class 'mixmeta'
coef(object, format=c("vector","matrix"), ...)

## S3 method for class 'mixmeta'
vcov(object, ...)
```

Arguments

object	an object of class "mixmeta".
format	format of the returned object.
...	further arguments passed to or from other methods.

Value

For coef, by default a vector (default)with the estimated fixed-effects coefficients. For multivariate models, a matrix can also be returned.

For vcov, the (co)variance matrix of the estimated fixed-effects coefficients.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See mixmeta-package for an overview of the package and modelling framework.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98)

# COEFFICIENTS
model$coef
coef(model)
coef(model, format="matrix")
summary(model)$coef

# (CO)VARIANCE MATRIX
vcov(model)
```

---

dbs *Deep-Brain Stimulation for Patients with Parkinson's Disease*

Description

The dataset contains the data on 46 studies published between 1980 and 2004 that assessed the effect of deep-brain stimulation on the relief of symptoms of Parkinson's disease. The outcome is reported as a score motor function, defined with the Unified Parkinson's Disease Rating Scale (UPDRS-part III), with lower values indicating better prognosis. Changes in the score were measured at 3, 6, 12 months and long-term after the implantation of the stimulator.

Usage

```
dbs
```

Format

A data frame with 68 observations on the following 12 variables:

- `author`: label identifying the study.
- `year`: year of publication.
- `eff_month3`, `var_month3`: point estimate and variance of the change in the score at 3 months.
- `eff_month6`, `var_month6`: point estimate and variance of the change in the score at 6 months.
- `eff_month12`, `var_month12`: point estimate and variance of the change in the score at 12 months.
- `eff_long`, `var_long`: point estimate and variance of the change in the score in the long term.
- `duration`: average disease duration (years).
- `baseline`: average baseline score of the patients.

Details

The data are stored in a *wide* format, with each record belonging to a single study and different variables providing estimates of the outcome at different times. Each study report results at one or multiple times, with the remaining times set to missing. See the dataset gliomas for an example of similar dataset stored in *long* format.

Note

The data provide an example of application of longitudinal meta-analysis, with repeated measurements of the effect size taken at various time point within each study. This requires a modelling structure that accounts for both within and between-study correlations of repeated measurements. In this case, the analysis is performed in the wide-format dataset using a multivariate meta-analysis. However, a long format is better suited for longitudinal meta-analysis, as it is applicable even when estimates are reported at different times in each study (see the examples in the help page of the dataset `gliomas`). Results can be compared with those reported Ishak and colleagues (2007).

Source

Ishak KJ, et al (2007). Meta-analysis of effect sizes reported at multiple time points using general linear mixed model. *Clinical Trials*. **4**(5):525–39.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN ISHAK ET AL (2007), TABLES 1 AND 2

# CREATE THE OUTCOME AND WITHIN-STUDY MATRICES (THE LATTER WITHOUT CORRELATION)
y <- as.matrix(dbs[1:4*2+1])
S <- as.matrix(dbs[1:4*2+2])

# INDEPENDENT RANDOM EFFECTS (TABLE 1, FIRST MODEL)
mv1 <- mixmeta(y ~ 1, S, bscov="diag", data=dbs)
```

```
print(summary(mv1), digits=1, report="var")

# HETEROGENEOUS AR1 RANDOM-EFFECTS (TABLE 1, THIRD MODEL)
mv3 <- mixmeta(y ~ 1, S, bscov="har1", data=dbs)
print(summary(mv3), digits=1, report="var")

# BUILD THE LIST HETEROGENEOUS AR1 WITHIN-STUDY ERRORS (CORRELATION AT 0.97)
cormat <- 0.97^abs(col(matrix(1,4,4)) - row(col(matrix(1,4,4))))
addS <- lapply(seq(nrow(S)), function(i) inputcov(sqrt(S[i,]), cormat))
addS <- lapply(addS, function(x) x[apply(!is.na(x),1,any), apply(!is.na(x),2,any)])

# ADD HAR1 WITHIN-STUDY ERRORS (TABLE 1, FOURTH MODEL) USING addSlist
## Not run:
mv4 <- mixmeta(y ~ 1, bscov="har1", data=dbs, control=list(addSlist=addS))
print(summary(mv4), digits=1, report="var")
## End(Not run)

## Not run:
### USE A LONG FORMAT, AS MORE FLEXIBLE AND ALLOWS MORE COMPLEX MODELS

# RESHAPE THE DATASET
long <- reshape(dbs, direction="long", idvar="author", v.names=c("eff","var"),
  varying=list(1:4*2+1, 1:4*2+2))

# RE-RUN THE LAST (FOURTH) MODEL
mv4b <- mixmeta(eff ~ factor(time) - 1, random = ~ factor(time) -1 | author,
  bscov="har1", data=long, control=list(addSlist=addS))
print(summary(mv4b), digits=1, report="var")

# COMMON RANDOM EFFECTS (TABLE 1, SECOND MODEL)
mv2 <- mixmeta(eff ~ factor(time) - 1, var, random = ~ factor(time) -1 | author,
  bscov="id", data=long)
print(summary(mv2), digits=1, report="var")

# FOURTH MODEL WITH ADDITIONAL CENTERED META-PREDICTORS (TABLE 2)
mv4plus <- mixmeta(eff ~ factor(time) - 1 + I(duration-14) + I(baseline-52),
  random = ~ factor(time) -1 | author, bscov="har1", data=long,
  control=list(addSlist=addS))
print(summary(mv4plus), digits=1, report="var")
## End(Not run)


### SEE help(gliomas) FOR A COMPLEMENTARY EXAMPLE
```

`fibrinogen` *Fibrinogen Studies Collaboration*

Description

The Fibrinogen Studies Collaboration is a meta-analysis of individual data on 154,012 adults from 31 prospective cohort studies with information on plasma fibrinogen and major disease outcomes.

The dataset reports a subset of the results of a first-stage analysis consisting of the log-hazard ratio of coronary heart disease for categories of levels of fibrinogen versus a baseline category.

Usage

```
fibrinogen
```

Format

A data frame with 31 observations on the following 15 variables:

- `cohort`: study ID.
- `b2, b3, b4, b5`: estimated log-hazard ratios for the second to fifth categories versus the baseline category.
- `V_2_2, V_3_3, V_4_4, V_5_5`: variances of the estimated log-hazard ratios.
- `V_2_3, V_2_4, V_2_5, V_3_4, V_3_5, V_4_5`: covariances of the estimated log-hazard ratios.

Details

The published analysis adopted a fixed-effects model on 10 categories of fibrinogen (Fibrinogen Studies Collaboration 2004, 2005). Here a subset of the results of the first-stage analysis is reported, namely the log-hazard ratio for 4 categories and associated (co)variance terms, ordered as the lower triangular elements of the (co)variance matrix taken by column. Details on the first-stage model and the second-stage meta-analysis are provided in White (2009) and Jackson and colleagues (2010).

Note

The data provide an example of application of multivariate meta-analysis for multi-parameter association, where a relationship is defined by functions specified by several coefficients. In this case, the coefficients refer to log-hazard ratio for strata of the original variable versus a baseline category. A general overview of the application of multivariate meta-analysis in this setting is provided by Gasparrini and colleagues (2012).

Source

Fibrinogen Studies Collaboration (2004). Collaborative meta-analysis of prospective studies of plasma fibrinogen and cardiovascular disease. *European Journal of Cardiovascular Prevention and Rehabilitation*. **11**:9–17.

Fibrinogen Studies Collaboration (2005). Plasma fibrinogen level and the risk of major cardiovascular diseases and nonvascular mortality: an individual participant meta-analysis. *Journal of the American Medical Association*. **294**:1799–1809.

White IR (2009). Multivariate random-effects meta-analysis. *Stata Journal*. **9**(1):40–56.

Jackson D, White IR, Thompson SG (2010). Extending DerSimonian and Laird's methodology to perform multivariate random effects meta-analyses. *Statistics in Medicine*. **29**(12):1282–1297.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN WHITE (2009) AND JACKSON ET AL. (2010)

# INSPECT THE DATA
head(fibrinogen)

# REML MODEL
y <- as.matrix(fibrinogen[2:5])
S <- as.matrix(fibrinogen[6:15])
model <- mixmeta(y, S)

# SUMMARIZE THE RESULTS
print(summary(model), digits=3)
round(model$Psi, 3)
```

`gliomas` *Randomized Trials on Therapies for Malignant Gliomas*

Description

The dataset contains the data on 17 randomized controlled trials comparing post-operative radiation therapy plus chemotherapy versus radiation therapy alone in patients with malignant gliomas. The outcome of interest is the probability of survival along time, measured as odds ratio at 6, 12, 18, and 24 months.

Usage

```
gliomas
```

Format

A data frame with 68 observations on the following 8 variables:

- `study`: number identifying the trial.
- `time`: time (months) since the start of the treatment at which survival status is assessed.
- `ntreat, streat`: number of total patients at the beginning of the study and surviving patients at specific times, respectively, in the treatment group (radiation therapy plus chemotherapy).
- `dcontr, ncontr`: number of total patients at the beginning of the study and surviving patients at specific times, respectively, in the control group (radiation alone).
- `logOR, varOR`: log-odds ratio of survival between treatment and control groups.

Details

The data are stored in a *long* format, with each record providing the estimate at a single time and each study providing multiple records. There were missing data for study 17 at months 6 and 18. There were no survivors in the control group at month 24 for studies 3 and 10, although this still

allows computation of the OR. See the dataset dbs for an example of similar dataset stored in *wide* format.

The log-odds ratio is computed empirically as $\log(s_t \times (n_t - s_t)/((n_c - s_c) \times s_c)$. Its variance is simply computed as $1/s_t + 1/(n_t - s_t) + 1/(n_c - s_c) + 1/s_c$.

Note

The data provide an example of application of longitudinal meta-analysis, with repeated measurements of the effect size taken at various time points within each study. This requires a modelling structure that accounts for both within and between-study correlations of repeated measurements. In this case, the same analysis can be performed in a wide-format dataset using a multivariate meta-analysis (see the examples in the help page of the dataset dbs). However, the long format is better suited for longitudinal meta-analysis, as it is applicable even when estimates are reported at different times in each study. Results can be compared with those reported Musekiwa and colleagues (2016). The same dataset was also used by Trikalinos and Olkin (2012), using a similar modelling scheme.

Source

Musekiwa A, et al (2012). Meta-analysis of effect sizes reported at multiple time points using general linear mixed model. *Plos One*. **11**(10):e0164898.

Trikalinos TA, Olkin I (2012). Meta-analysis of effect sizes reported at multiple time points: a multivariate approach. *Clinical Trials*. **9**(5):610–620.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN MUSEKIWA ET AL (2012), TABLES 3 AND 4

# INDEPENDENT RANDOM EFFECTS, NO WITHIN-STUDY CORRELATION (MODEL 1)
mod1 <- mixmeta(logOR~0+factor(time), S=logORvar, random=~0+factor(time)|study,
  bscov="diag", data=gliomas)
print(summary(mod1), digits=2, report="var")

# COMPOUND-SYMMETRY RANDOM EFFECTS, NO WITHIN-STUDY CORRELATION (MODEL 2)
# NB: THIS REQUIRES A TWO-LEVEL MODEL WITH THE INNER-LEVEL VARIANCE FIXED TO 0
unit <- factor(seq(nrow(gliomas)))
mod2 <- mixmeta(logOR~0+factor(time), S=logORvar, random=~1|study/unit,
  bscov=c("unstr","fixed"), data=gliomas, control=list(Psifix=list(unit=0)))
print(summary(mod2), digits=2, report="var")

# BUILD THE HETEROGENEOUS AR1 WITHIN-STUDY ERRORS (CORRELATION AT 0.61)
cormat <- 0.61^abs(col(matrix(1,4,4)) - row(col(matrix(1,4,4))))
addS <- lapply(split(sqrt(gliomas$logORvar), gliomas$study), inputcov, cormat)
addS <- lapply(addS, function(x) x[apply(!is.na(x),1,any), apply(!is.na(x),2,any)])

# INDEPENDENT RANDOM EFFECTS, HAR1 WITHIN-STUDY CORRELATION (MODEL 4)
mod4 <- mixmeta(logOR~0+factor(time), random=~0+factor(time)|study,
  bscov="diag", data=gliomas, control=list(addSlist=addS))
```

```
print(summary(mod4), digits=2, report="var")

# UNSTRUCTURED RANDOM EFFECTS, HAR1 WITHIN-STUDY CORRELATION (MODEL 6)
mod6 <- update(mod4, bscov="unstr")
print(summary(mod6), digits=2, report="var")

# COMPARE THE FIT
AIC(mod1, mod2, mod4, mod6)

## Not run:
### MORE FLEXIBLE MODELLING OF RANDOM EFFECTS

# RE-RUN BEST FITTING MODEL WITH ML (ALLOWS TESTING OF FIXED EFFECTS)
mod4ml <- update(mod4, method="ml")

# RANDOM-SLOPE MODEL WITH TIME AS CONTINUOUS (CENTERED IN random)
modnew <- mixmeta(logOR~time, random=~I(time-15)|study, bscov="diag",
  method="ml", data=gliomas, control=list(addSlist=addS, maxiter=200))
print(summary(modnew), digits=2, report="var")

# COMPARE
AIC(mod4ml, modnew)
## End(Not run)


### SEE help(dbs) FOR A COMPLEMENTARY EXAMPLE
```

`hsls` *High School Longitudinal Study*

Description

This is a nationally representative, longitudinal study of more than 21,000 9th graders in 944 schools who will be followed through their secondary and postsecondary years. The data are used for testing whether sex, socioeconomic status and sex by socio-economic status interaction are predictive of the mathematics standardized score in each of the eight race groups.

Usage

```
hsls
```

Format

A data frame with 8 observations on the following 10 variables:

- `race`: race group.
- `b1`, `b2`, `b3`: estimated regression coefficients for sex, socio-economic status and sex by socio-economic status interaction, respectively, on the mathematics standardized score.
- `V11`, `V22`, `V33`: variances of the estimated coefficients.
- `V12`, `V13`, `V23`: covariances of the estimated coefficients.

Source

Chen H, Manning AK, Dupuis J (2012). A method of moments estimator for random effect multivariate meta-analysis. *Biometrics*. **68**(4):1278–1284.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN CHEN ET AL. (2012)

# INSPECT THE DATA
hsls

# FIXED-EFFECTS MODEL
S <- as.matrix(hsls[5:10])
mod1 <- mixmeta(cbind(b1,b2,b3), S, data=hsls, method="fixed")
summary(mod1)

# MM MODEL
mod2 <- mixmeta(cbind(b1,b2,b3), S,data=hsls, method="mm")
summary(mod2)
mod2$Psi
```

---

hyp *Ten Studies Assessing an Hypertension Treatment*

Description

The dataset contains the results of ten studies that assess the effectiveness of hypertension treatment for lowering blood pressure. Each study provides complete data on two treatment effects, the difference in systolic blood pressure (SBP) and diastolic blood pressure (DBP) between the treatment and the control groups, where these differences are adjusted for the participants' baseline blood pressures. The within-study correlations of the two outcomes are known. Some trials are conducted on patients with isolated systolic hypertension (ISH).

Usage

```
hyp
```

Format

A data frame with 10 observations on the following 7 variables:

- `study`: study ID.
- `sbp, sbp_se`: estimated difference and its standard error in systolic blood pressure.
- `dbp, dbp_se`: estimated difference and its standard error in diastolic blood pressure.
- `rho`: within-study correlation between the estimated differences in systolic and diastolic blood pressure.
- `ish`: indicator for studies on patients with isolated systolic hypertension.

Note

The standard errors for the two outcomes are wrongly reported as variances in the original article by Jackson and colleagues (2013).

Source

Jackson D, White IR, Riley RD (2013). A matrix based method of moments for fitting the multivariate random effects model for meta-analysis and meta-regression. *Biometrical Journal*. **55**(2):231–45.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN JACKSON ET AL. (2013)

# INSPECT THE DATA
hyp

# INPUT THE CORRELATION (CAN ALSO BE INPUTTED DIRECTLY, SEE BELOW)
(S <- inputcov(hyp[c("sbp_se", "dbp_se")], cor=hyp$rho))
# CHECK WITH THE FIRST STUDY
cov2cor(xpndMat(S[1,]))

# META-ANALYSIS, REML MODEL
mod1 <- mixmeta(cbind(sbp,dbp), S=S, data=hyp)
print(summary(mod1), digits=2)
round(mod1$Psi,2)

# META-ANALYSIS, REML MODEL (INPUTTING THE CORRELATION DIRECTLY)
mod2 <- mixmeta(cbind(sbp,dbp), S=cbind(sbp_se,dbp_se)^2, data=hyp,
  control=list(Scor=hyp$rho))
print(summary(mod2), digits=2)

# META-ANALYSIS, MM MODEL
mod3 <- mixmeta(cbind(sbp,dbp), S=S, data=hyp, method="mm")
print(summary(mod3), digits=2)
round(mod3$Psi,2)

# META-REGRESSION, REML MODEL
mod4 <- mixmeta(cbind(sbp,dbp) ~ ish, S=S, data=hyp)
print(summary(mod4), digits=2)

# META-REGRESSION, MM MODEL
mod5 <- mixmeta(cbind(sbp,dbp) ~ ish, S=S, data=hyp, method="mm")
print(summary(mod5), digits=2)
```

`inputcov` *Input (Co)Variance Matrices*

Description

This function inputs (co)variance matrices of a set of outcomes given the corresponding standard deviation and correlation values.

Usage

```
inputcov(sd, cor=NULL)
```

Arguments

`sd`	a $m \times k$ matrix of standard deviations for k outcomes in m matrices, or a vector for k outcomes in a single matrix.
`cor`	either a vector of length 1, m or $k(k-1)/2$, or alternatively a $k \times k$ or $m \times k(k-1)/2$ matrix. See Details.

Details

Depending the number of outcomes k and matrices m, the argument `cor` is interpreted as:

- if a vector of length 1 (a scalar), the same correlation for all the k outcomes for all the m matrices;
- if a vector of length m, the same correlation for all the k outcomes for each of the m matrices;
- if a vector of length $k(k-1)/2$, the lower triangular elements (without diagonal, taken by column) of the correlation matrix of the k outcomes, the same for all the m matrices;
- if a $k \times k$ matrix, the correlation matrix for the single matrix (only when m=1);
- if a $m \times k(k-1)/2$ matrix, each row represents the lower triangular elements (without diagonal, taken by column) of the correlation matrix of the k outcomes for each of the m matrices.

Value

For a single matrix, the (co)variance matrix itself. For multiple matrices, a $m \times k(k+1)/2$ matrix, where each row represents the vectorized entries of the lower triangle (with diagonal, taken by column) of the related (co)variance matrix (see `vechMat`).

Note

This function is called internally by `mixmeta` for multivariate models to input the correlation(s) when only the within-unit variances are provided through the argument `S`. In this case, the correlation values are set through the argument `Scor` in the control list (see `mixmeta.control`).

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See xpndMat. See mixmeta.control.

Examples

```
# SOME RANDOM SD FOR A SINGLE MATRIX, WITH CONSTANT CORRELATION
(M <- inputcov(runif(4, 0.1, 3), 0.7))
# CHECK CORRELATION
cov2cor(M)

# NOW WITH A MORE COMPLEX CORRELATION STRUCTURE
(M <- inputcov(runif(3, 0.1, 3), c(0.7,0.2,0.4)))
cov2cor(M)

# MULTIPLE MATRICES
(V <- matrix(runif(5*3, 0.1, 3), 5, 3,
  dimnames=list(1:5, paste("V", 1:3, sep=""))))
inputcov(V, 0.6)

# WITH REAL DATA WHEN CORRELATIONS AVAILABLE
hyp
(S <- inputcov(hyp[c("sbp_se","dbp_se")], cor=hyp$rho))
# CHECK FIRST STUDY
cov2cor(xpndMat(S[1,]))

# USED INTERNALLY IN mixmeta
p53
inputcov(sqrt(p53[c("V1","V2")]), 0.5)
model <- mixmeta(cbind(y1,y2), S=cbind(V1,V2), data=p53, control=list(Scor=0.5))
model$S
```

---

inputna *Input Missing Values*

Description

This function augment data by replacing missing values. It can be used internally in mixmeta through the control list.

Usage

```
inputna(y, S, inputvar=10^4)
```

Arguments

Assuming a meta-analysis or meta-regression based on n units and k outcomes:

y	a n-dimensional vector (for univariate models) or $m \times k$ matrix (for multivariate models) of outcomes.

S	series of within-unit variances (or (co)variance matrices for multivariate models) of the estimated outcome(s). For univariate models, this is usually a n-dimensional vector. For multivariate models, it can be provided as: a m-dimensional list of $k \times k$ matrices; a tri-dimensional $k \times k \times m$ array; a matrix or data frame with n rows and $k(k+1)/2$ or k columns, depending on the availability of the within-unit correlations.
inputvar	multiplier for inputting the missing variances in S.

Details

The function augments the data by replacing missing values in the outcomes and the associated (co)variances. Specifically, it replaces missing outcomes and missing covariances (if provided) with 0, and missing variances with the largest observed variance multiplied by inputvar. This value is expected to be very high, by default 10^4, so that the corresponding observation contributes only negligibly to the final estimate.

Value

A matrix with the first k column corresponding to the augmented outcomes, and the remaining $k(k+1)/2$ or k columns (depending on the availability of the within-study covariances) corresponding to vectorized entries of the lower triangle of the related (co)variance matrices.

Note

Data augmentation used to be the approach to deal with missing values in the first implementation of **mixmeta**. The current algorithms directly account for missing.

Inputting missing values can be useful when two or more outcomes are never observed jointly, and the estimation is entirely based on indirect comparison. This method can be applied in network meta-analysis, also called indirect treatment comparison.

This approach can produce different results than standard methods, especially when the occurrence of missing is substantial. Preliminary analyses indicate that likelihood-based estimation methods do not seem to be affected, while non-iterative estimators such as method of moments and variance components are more sensitive. The user should be careful on the application of missing augmentation.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

Jackson D, Riley R, White IR (2011). Multivariate meta-analysis: Potential and promise. *Statistics in Medicine*. **30**(20);2481–2498.

White IR (2009). Multivariate random-effects meta-analysis. *Stata Journal*. **9**(1):40–56.

White IR (2011). Multivariate random-effects meta-regression: updates to mvmeta. *Stata Journal*. **11**(2):255-270.

See Also

See `inputcov` for inputting (co)variance matrices.

Examples

```
# INSPECT THE DATA
head(smoking)

# STANDARD APPROACH TO MISSING DATA
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
mod1 <- mixmeta(y, S)
summary(mod1)

# WITH DATA AUGMENTATION
augdata <- inputna(y, S)
y <- augdata[,1:3]
S <- augdata[,-c(1:3)]
mod2 <- mixmeta(y, S)
summary(mod2)
# NB: SAME PARAMETER ESTIMATES, BUT WRONG NYUMBER OF OBS

# USED INTERNALLY IN mixmeta
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
mod3 <- mixmeta(y, S, control=list(inputna=TRUE))
summary(mod3)
# NOW RIGHT NUMBER OF OBS
```

---

logLik.mixmeta *Extract Log-Likelihood from mixmeta Objects*

Description

This method function returns the (restricted) log-likelihood for fitted meta-analytical models represented in objects of class `"mixmeta"`.

Usage

```
## S3 method for class 'mixmeta'
logLik(object, ...)
```

Arguments

object	an object of class "mixmeta".
...	further arguments passed to or from other methods.

Value

A numeric scalar of class "logLik" with attributes, providing the (restricted) log likelihood of the model. Attributes correspond to the component df of mixmeta objects, namely the following scalars: nall (number of observations used for estimation, excluding missing values), nobs (equal to nall, minus the number of fixed-effects coefficients for REML models, fixed (number of estimated fixed-effects coefficients), random (number of estimated (co)variance terms).

Note

This functions is called by AIC and BIC for computing the Akaike and Bayesian information criteria.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See the default method logLik. See mixmeta-package for an overview of the package and modelling framework.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL)~pubyear,S=berkey98[5:7],data=berkey98)

# LOG-LIKELIHOOD
ll <- logLik(model)
ll
attributes(ll)

# AIC and BIC
AIC(model)
BIC(model)
```

`mixmeta` *Fitting Standard and Extended Meta-Analysis and Meta-Regression Models*

Description

The function `mixmeta` performs various meta-analytical models under a common mixed-effects framework, including standard univariate fixed and random-effects meta-analysis and meta-regression, and non-standard extensions such as multivariate, multilevel, longitudinal, and dose-response models. The function `mixmeta.fit` is a wrapper for actual fitting functions based on different estimation methods, usually called internally. See `mixmeta-package` for an overview.

Usage

```
mixmeta(formula, S, data, random, method="reml", bscov="unstr", offset, subset,
  contrasts=NULL, na.action, model=TRUE, control=list())

mixmeta.fit(X, Z, y, S, groups, method, bscov, control)
```

Arguments

Assuming a meta-analysis or meta-regression based on n units aggregated within m (outer-level) groups, k outcomes, p fixed-effects predictors, and q random-effects predictors:

`formula` an object of class `"formula"` (or one that can be coerced to that class) offering a symbolic description of the linear predictor for the fixed-effects part of the model. Alternatively, for meta-analysis with no fixed-effects predictors, a single vector (for univariate models) or matrix-type object (for multivariate models). Terms in `formula` must be vector or matrix-type objects, optionally provided in the `data` argument below. See `mixmetaFormula`.

`S` series of within-unit variances (or (co)variance matrices for multivariate models) of the estimated outcome(s). For univariate models, this is usually a n-dimensional vector. For multivariate models, it can be provided as: a m-dimensional list of $k \times k$ matrices; a tri-dimensional $k \times k \times m$ array; a matrix or data frame with n rows and $k(k+1)/2$ or k columns, depending on the availability of the within-unit correlations. `mixmeta.fit` accepts only the last option. Optionally, for more complex error structures, this argument can be omitted and passed through `addSlist` in `control`. See Details below.

`data` an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in `formula` and `random`. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `mixmeta` is called.

`random` a one-sided formula (or a list of formulae for multilevel models) offering a symbolic description of the linear predictor(s) and grouping structure for the random-effects part of the model. The usual form is `~ z1 + ... + zq | g`, with the grouping factor separated from the linear predictor by the symbol `'|'`. Multiple levels with the same linear predictor can be defined by separating multiple

grouping factors using the symbol '/'. Alternatively, in a list form the grouping factors can also be provided as list names. In both cases, the levels are considered nested (from outer to inner following the order). See mixmetaFormula and Details below.

method — estimation method: "fixed" for fixed-effects models, "ml" or "reml" for random-effects models fitted through (restricted) maximum likelihood, "mm" for random-effects models fitted through method of moments, and "vc" for random-effects models fitted through variance components. See Details below. If "model.frame", the model frame is returned, as in lm or glm.

bscov — a character vector defining the structure of the random-effects (co)variance matrices. Default to "unstr" (unstructured). Names corresponding to grouping factors (see random above) can be used to refer to specific random-effects levels for non-default values. If unnamed, the values can be recycled. Among various (co)variance structures, the user can select "diag" (diagonal), "cs" (compound symmetry), "hcs" (heterogeneous compound symmetry), "ar1" (autoregressive of first order), or "fixed" (fixed). See also Details.

offset — optionally, a n-dimensional numeric vector used to specify an a priori known component in the linear predictor. One or more offset terms can be included in the formula instead or as well. See model.offset.

subset — an optional vector specifying a subset of observations to be used in the fitting process.

contrasts — an optional list. See the contrasts.arg of model.matrix.

na.action — a function which indicates what should happen when the data contain NAs. Default to na.action setting of options, usually na.omit. na.exclude can be useful. See details on missing values in mixmeta.

model — a logical value indicating whether the model frame should be included as a component of the returned value. See the model.frame method function.

control — list of parameters for controlling the fitting process. These are passed to mixmeta.control to replace otherwise selected default values.

X — a $n \times p$ design matrix containing the p fixed-effects predictors, appropriately ordered by groups. Usually produced internally by mixmeta from formula above.

Z — a $n \times q$ design matrix (or a list of design matrices for multilevel models) containing the q random-effects predictors, appropriately ordered by groups. Usually produced internally by mixmeta from random above.

y — a n-dimensional vector (for univariate models) or $m \times k$ matrix (for multivariate models) of outcomes, appropriately ordered by groups. Usually produced internally by mixmeta from formula above.

groups — matrix with n rows, with each column identifying the groups for each level of random-effects, appropriately ordered. Usually produced internally by mixmeta from random above.

Details

The function mixmeta resembles standard regression functions in R. See lme in particular, or lm or glm, for information on most of the arguments. Internally, this function assembles the data components, defines the (potentially multiple) grouping levels and re-order the data accordingly, and

then pass them to `mixmeta.fit`. This is a wrapper for actual fitting functions that are automatically selected. Functions other than `mixmeta` are not expected to be called directly for model fitting.

Fixed or random-effects models for meta-analysis are simply defined using `y ~ 1` in `formula`, where `y` is a response vector optionally stored in `data`. In meta-regression models, other terms are added in the right-hand side of the formula as `y ~ x1 + ... + xp`, defining the linear meta-predictor. Factors, variable transformations and interactions are allowed, following the usual formula specification (see `mixmetaFormula`).

In this standard usage, each of the n rows is assumed to represent a single estimate of an outcome from a set of independent studies. In random-effects models, the grouping structure is automatically derived by assigning a group to each row of data (with $m = n$). Extensions to multivariate models ($k > 1$) are straightforward, and only require using a matrix in the left-hand side, where each of the k columns represents a different outcome, or the form `cbind(y1, ..., yk) ~ 1`. See `mixmetaFormula`.

Non-standard random-effects models can be specified through the optional argument `random`. This is commonly represented by a one-sided formula, whose basic random-intercept form is `~ 1 | g`, where `g` is a grouping factor. A more complex linear meta-predictor for the random-effects part can be also specified by `~ z1 + ... + zq | g`. The argument `random` also accepts a list of one-sided formulae referring to multiple random-effects levels (see `mixmetaFormula`). The use of `random` extends the standard meta-analytical setting by relaxing the assumption of independence between units, allowing multiple estimates from the same group (with $m < n$) and multiple nested grouping levels. This provides the possibility to fit longitudinal, multilevel, and dose-response meta-analysis, among other extensions. See the examples below.

The argument `bscov` allows the definition of specific structures for the random-effects (co)variance matrices corresponding the each level. The default unstructured form requires `kq(kq+1)/2` parameters for a single-level meta-analysis. The choice of other structures reduces the number of parameters, although requiring stronger assumptions. More information and complete list of options is available at a specific help page (see `mixmetaCovStruct`).

The within-unit (co)variances are provided through the argument `S`, usually as a matrix. If the correlations are available, each of the m row represents the $k(k+1)/2$ vectorized entries of the lower triangle of the related (co)variance matrix, taken by column (see `xpndMat`). If correlations are not available, each row represents the k variances, and the correlations are inputted internally through the argument `Scor` of the `control` list (see `inputcov`). For more complex error structures that span multiple units, the argument `S` can be omitted and passed through `addSlist` in `control`, although requiring the observations to be re-ordered accordingly to groups (see `mixmeta.control`).

Different estimator are available in the package `mixmeta` and chosen through the argument `method`, with related fitting functions called internally. In the current version, the options are:

- `method="fixed"`: `Fixed-effects estimator`
- `method="ml"`: `Maximum likelihood (ML) estimator`
- `method="reml"`: `Restricted maximum likelihood (REML) estimator`
- `method="mm"`: `Method of moments estimator`
- `method="vc"`: `Variance components estimator`

Note that non-standard random-effects models and the use of structured (co)variance matrices are only available for `"ml"` and `"reml"` methods. See their help pages for further details on the estimation procedures, following the links above.

Missing values are allowed in both sides of formula. In the case of missing predictors (right-hand side of formula), the related unit is entirely excluded from estimation. In contrast, a unit still contributes to estimation if at least outcome is non-missing. This behaviour is different from standard regression functions such as lm or glm. Before the call to mixmeta.fit, units matching such stricter missing definition are removed from the the model frame. The missing pattern in S must be consistent with that in y. See further details on handling missing values in mixmeta.

The fitting procedure can be controlled through the additional terms specified in control, which are passed to the function mixmeta.control.

Value

The mixmeta function typically returns a list object of class "mixmeta" representing the meta-analytical model fit, as described in mixmetaObject. When method="data.frame", the model is not fitted and the model frame is returned, namely a data frame with special attributes (see the default method model.frame) and, in this case, the additional class "data.frame.mixmeta".

The wrapper function mixmeta.fit is usually called internally in mixmeta, and returns an intermediate list object with some of the components expected in the "mixmeta" class.

Several method functions for regression objects are available, either default or specifically written for the "mixmeta" class. See mixmetaObject for a complete list.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Gasparrini A. (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

See Also

See additional info on the estimation procedures at the related page of the fitting functions See mixmetaFormula for the use of formulae to define the fixed and random parts of the model. See alternative (co)variance structures for likelihood-based estimation methods. See handling of missing values in mixmeta. See lme, lm or glm for standard regression functions. See mixmeta-package for an overview of this modelling framework.

Examples

```
### STANDARD MODELS

# RANDOM-EFFECTS META-ANALYSIS, ESTIMATED WITH REML
model <- mixmeta(logor, logorvar, data=bcg)
summary(model)

# RANDOM-EFFECTS META-REGRESSION, ESTIMATED WITH ML
```

```
model <- mixmeta(logor~ablat, logorvar, data=bcg, method="ml")
summary(model)


### MAIN METHOD FUNCTIONS

# COEFFICIENTS AND (CO)VARIANCE MATRIX
coef(model)
vcov(model)

# RESIDUALS AND FITTED VALUES
residuals(model)
fitted(model)

# MODEL FRAME AND MODEL MATRIX
model.frame(model)
model.matrix(model)

# LOG-LIKELIHOOD AND AIC VALUE
logLik(model)
AIC(model)

# COCHRAN Q TEST FOR RESIDUAL HETEROGENEITY
qtest(model)


### PREDICTIONS

# PREDICTED EFFECTS
predict(model)
predict(model, se=TRUE)
predict(model, newdata=data.frame(ablat=2:5*10), ci=TRUE)

# BEST LINEAR UNBIASED PREDICTION
blup(model)
blup(model, pi=TRUE)

# SEE help(predict.mixmeta) AND help(BLUP.mixmeta) FOR MORE INFO


### MULTIVARIATE MODELS

### BIVARIATE MODELS
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98)
summary(model)
residuals(model)

### MULTIVARIATE META-ANALYSIS WITH MORE THAN 2 OUTCOMES
y <- as.matrix(fibrinogen[2:5])
S <- as.matrix(fibrinogen[6:15])
model <- mixmeta(y, S)
summary(model)
predict(model, se=TRUE)
```

```
predict(model, se=TRUE, aggregate="outcome")


### OTHER EXTENSIONS

# MULTILEVEL META-ANALYSIS
model <- mixmeta(effect, var, random= ~ 1|district/study, data=school)
summary(model)
# SEE help(school) AND help(thrombolytic) FOR MORE EXAMPLES

# DOSE-RESPONSE META-ANALYSIS (SIMPLIFIED)
model <- mixmeta(logrr ~ 0 + dose, S=se^2, random= ~ 0 + dose|id, data=alcohol,
 subset=!is.na(se))
summary(model)
# SEE help(alcohol) FOR MORE EXAMPLES

# LONGITUDINAL META-ANALYSIS
model <- mixmeta(logOR~time, S=logORvar, random=~I(time-15)|study, data=gliomas)
summary(model)
# SEE help(gliomas) AND help(dbs) FOR MORE EXAMPLES


### FIXED-EFFECTS MODELS AND ALTERNATIVE ESTIMATORS

# FIXED-EFFECTS MODEL
model <- mixmeta(sbp~ish, S=sbp_se^2, data=hyp, method="fixed")
summary(model)

# METHOD OF MOMENTS
S <- as.matrix(hsls[5:10])
model <- mixmeta(cbind(b1,b2,b3), S, data=hsls, method="mm")
summary(model)

# VARIANCE COMPONENTS ESTIMATOR
model <- mixmeta(cbind(PD,AL)~pubyear, S=berkey98[5:7], data=berkey98,
  method="vc")
summary(model)


### IN THE PRESENCE OF MISSING VALUES

# RUN THE MODEL
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
model <- mixmeta(y, S)
summary(model)
model.frame(model)

# SEE help(na.omit.data.frame.mixmeta) FOR MORE EXAMPLES


### WHEN WITHIN-STUDY COVIARIANCES ARE NOT AVAILABLE AND/OR NEED TO BE INPUTTED
```

```
# GENERATE S
(S <- inputcov(hyp[c("sbp_se","dbp_se")], cor=hyp$rho))

# RUN THE MODEL
model <- mixmeta(cbind(sbp,dbp), S=S, data=hyp)

# INPUTTING THE CORRELATION DIRECTLY IN THE MODEL
model <- mixmeta(cbind(y1,y2), cbind(V1,V2), data=p53, control=list(Scor=0.95))
summary(model)

# SEE help(hyp) AND help(p53) FOR MORE EXAMPLES


### STRUCTURING THE BETWEEN-STUDY (CO)VARIANCE

# DIAGONAL
S <- as.matrix(hsls[5:10])
model <- mixmeta(cbind(b1,b2,b3), S, data=hsls, bscov="diag")
summary(model)
model$Psi

# COMPOUND SYMMETRY
model <- mixmeta(cbind(b1,b2,b3), S, data=hsls, bscov="cs")
summary(model)
model$Psi

# SEE help(mixmetaCovStruct) FOR DETAILS AND ADDITIONAL EXAMPLES


### USE OF THE CONTROL LIST

# PRINT THE ITERATIONS AND CHANGE THE DEFAULT FOR STARTING VALUES
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
model <- mixmeta(y, S, control=list(showiter=TRUE, igls.inititer=20))

# SEE help(mixmeta.control) FOR FURTHER DETAILS
```

`mixmeta.control` *Ancillary Parameters for Controlling the Fit in mixmeta Models*

Description

This internal function sets the parameter options used for fitting meta-analytical models, commonly to pre-specified default values. It is usually internally called by `mixmeta`.

Usage

```
mixmeta.control(optim=list(), showiter=FALSE, maxiter=100, initPsi=NULL, Psifix=NULL,
  Scor=NULL, addSlist=NULL, inputna=FALSE, inputvar=10^4, loglik.iter="hybrid",
```

```
  igls.inititer=10, hessian=FALSE, vc.adj=TRUE, reltol=sqrt(.Machine$double.eps),
   checkPD=NULL, set.negeigen=sqrt(.Machine$double.eps))
```

Arguments

optim	list of parameters passed to the control argument of the function optim, which performs the quasi-Newton optimization in likelihood-based random-effects models. See optim for the list of arguments. See Details for additional info.
showiter	logical. If TRUE, the progress of iterative optimization is shown.
maxiter	positive interger value. Maximum number of iterations in methods involving optimization procedures.
initPsi	either a matrix or a vector of its lower triangular elements (with diagonal, taken by column), or optionally a named list with one or more of such objects. Used as starting values of random-effects parameters in likelihood-based optimization routines. See Details.
Psifix	either a matrix or a vector of its lower triangular elements (with diagonal, taken by column), or optionally a named list with one or more of such objects. Used to define fixed parts of the random-effects (co)variance structures. See Details.
Scor	either a scalar, vector or matrix representing the within-unit correlation(s) to be inputted when the covariances are not provided in multivariate models, and ignored if they are. See inputcov.
addSlist	a list of m matrices for the (outer-level) groups of units defining the (known) error (co)variance structure, when this cannot be passed through the argument S of mixmeta. See Details.
inputna	logical. If missing values must be internally inputted. To be used with caution. See inputna.
inputvar	multiplier for inputting the missing variances, to be passed as an argument to inputna.
loglik.iter	iterative scheme used in in likelihood-based optimization routines. Options are "hybrid", "newton", and "igls" or "rigls". See mixmeta.ml.
igls.inititer	number of iterations of the (restricted) iterative generalized least square algorithm when used in the initial phase of hybrid optimization procedure of likelihood-based estimators. See mixmeta.ml.
hessian	logical. If TRUE, the Hessian matrix of the parameters estimated in the optimization process is computed and returned. Only applicable to likelihood-based estimation methods. For details, see the info provided in the help pages of the optimizations algorithms and (co)variance structures.
vc.adj	logical. If TRUE, an adjustement to the way the marginal variance part is computed in the (co)variance components estimator is applied in the variance components estimator. See mixmeta.vc.
reltol	relative convergence tolerance in methods involving optimization procedures. The algorithm stops if it is unable to reduce the value by a factor of reltol * (abs(val) + reltol) at a step.

checkPD	logical. Determines if the semi-positiveness of within-unit error or random-effects (co)variance matrices must be checked.
set.negeigen	positive value. Value to which negative eigenvalues are to be set in estimators where such method is used to force semi-positive definiteness of the estimated between-study (co)variance matrix.

Details

This function has default values for most of the arguments, some of them set internally. Non-default values are passed through the control argument of mixmeta. Many arguments refer to specific fitting procedures. See the help page of the related estimator for details.

The function automatically sets non-default values for some control arguments for optim, unless explicitly set in the list passed to it. Specifically, the function selects fnscale=-1, maxit=maxiter and reltol=reltol, where the latter two are specified by other arguments of this function.

The arguments initPsi and Psifix are used to provide information for estimation procedures of the random-effects parameters in likelihood-based methods. Specifically, the former is used to choose non-default starting values (see mixmeta.ml), and the latter for defining the fixed (known) part of specific (co)variance structures. In multilevel models, these arguments must be lists with named components referring to one or more levels of grouping defined by the argument random of mixmeta.

The argument addSlist can be used to define more complex (known) error structures of the outcome(s) that are usually provided through the argument S of mixmeta as within-unit variances (or (co)variance matrices for multivariate models). This can be useful when these error structures spans multiple units (rows), and the between-unit correlation cannot be defined through S, for instance in dose-response meta-analysis (see examples in mixmeta). Note that this information is passed internally after the data have be re-ordered following the grouping defined by random in mixmeta, and this should be consistent in addSlist. Specifically, the grouping variables are assumed as factors and therefore the groups are taken in alphabetical/numeric order. It is suggested to re-order the data according to this order of the groups before fitting the model, so to ensure consistency between the grouped data and addSlist.

Value

A list with components named as the arguments.

Note

The function is expected to be extended and/or modified at every release of the package **mixmeta**. It is strongly suggested to check the arguments of this function at every release.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See mixmeta. See also glm.control. See the help pages of the related fitting functions for details on each parameter. See mixmeta-package for an overview of this modelling framework.

Examples

```
# PRINT THE ITERATIONS (SEE ?optim) AND CHANGE THE DEFAULT FOR STARTING VALUES
mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98,
  control=list(showiter=TRUE, igls.inititer=20))

# INPUT THE CORRELATION
mixmeta(cbind(y1,y2), S=cbind(V1,V2), data=p53, control=list(Scor=0.5))

# FIX (PARTS OF) THE RANDOM-EFFECTS (CO)VARIANCE MATRIX
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
mixmeta(y, S, bscov="prop", control=list(Psifix=diag(3)+1))
```

---

mixmeta.fixed *Fixed-Effects Estimator for mixmeta Models*

Description

This function implements a generalized least square estimator for fixed-effects meta-analysis and meta-regression, including standard univariate models and non-standard multivariate extensions. It is meant to be used internally and not directly run by the users.

Usage

```
mixmeta.fixed(Xlist, ylist, Slist, nall, control, ...)
```

Arguments

Assuming a meta-analysis or meta-regression based on m independent groups (usually studies) providing a single estimate (unit of analysis), k outcomes and p fixed-effects predictors:

Xlist	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
ylist	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.
Slist	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
nall	numeric scalar with the total number of observations (excluding missing).
control	list of parameters for controlling the fitting process, usually internally set to default values by mixmeta.control.
...	further arguments passed to or from other methods. Currently not used.

Details

The estimation involves only the kp fixed-effects coefficients. Note that, in this fixed-effects estimator, each unit is assumed independent from the others, and therefore the number of groups (the length of the lists) is identical to the number of units (`m=n`). However, this is not important in fixed-effects models, where no random (and therefore grouping) structure is used.

The routine is based on a standard generalized least square (GLS) algorithm implemented in the internal function `glsfit`. The between-study (co)variance matrix is set to zero, so the marginal (co)variance matrix, composed only by elements of the within-unit component, is assumed as completely known. Similarly to the likelihood-based estimators implemented in `mixmeta.ml` and `mixmeta.reml`, the computation involves Cholesky and and QR decompositions for computational stability and efficiency. The method is described in details in Gasparrini and collaborators (2012) (see references below).

Value

These functions return an intermediate list object, with some components then processed and some others added later within `mixmeta.fit` and `mixmeta` to finalize an object of class `"mixmeta"`. See `mixmetaObject`.

Note

As stated earlier, this function is called internally by `mixmeta.fit`, and is not meant to be used directly. In particular, its code does not contain any check on the arguments provided, which are expected in specific formats. The function is however exported in the namespace and documented for completeness.

The arguments above are prepared by `mixmeta.fit` from its arguments `X`, `y` and `S`. The list structure, although requiring more elaborate coding, is computationally more efficient, as it avoids the specification of sparse block-diagonal matrices, especially for meta-analysis involving a large number of studies.

Some parameters of the fitting procedures are determined by the `control` argument, with default set by `mixmeta.control`. No missing values are accepted in the fitting functions. See details on `missing values`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

Berkey CS, Anderson JJ, Hoaglin DC (1996). Multiple-outcome meta-analysis of clinical trials. *Statistics in Medicine*. **15**(5):537–547.

Berkey CS, et al. (1998). Meta-analysis of multiple outcomes by regression with random effects. *Statistics in Medicine*. **17**(22):2537–2550.

See Also

See `mixmeta` for the general usage of the functions. See `mixmeta.control` to determine specific parameters of the fitting procedures. Use the triple colon operator (':::') to access the code of the internal functions, such as `glsfit`. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# UNIVARIATE FIXED-EFFECTS MODEL
mod1 <- mixmeta(yC, S=SCC, data=smoking, method="fixed")
summary(mod1)

# MULTIVARIATE FIXED-EFFECTS MODEL
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
mod2 <- mixmeta(y, S, method="fixed")
summary(mod2)

# MULTIVARIATE FIXED-EFFECTS MODEL: REPLICATE THE RESULTS IN BERKEY ET AL. 1998
mod3 <- mixmeta(cbind(PD,AL) ~ I(pubyear-1983), S=berkey98[5:7], data=berkey98,
  method="fixed")
summary(mod3)
```

`mixmeta.ml` *ML and REML Estimators for mixmeta Models*

Description

These functions implement maximum likelihood (ML) and restricted maximum likelihood (REML) estimators for random-effects meta-analysis and meta-regression, including standard univariate models, and non-standard extensions such as multivariate, multilevel, longitudinal, and dose-response models. These functions are meant to be used internally and not directly run by the users.

Usage

```
mixmeta.ml(Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, bscov, control, ...)

mixmeta.reml(Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, bscov, control, ...)
```

Arguments

Assuming a meta-analysis or meta-regression based on n units aggregated within m (outer-level) groups, k outcomes, p fixed-effects predictors, and q random-effects predictors:

`Xlist`	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
`Zlist`	a m-dimensional list of group-specific design matrices for the random-effects part of the model. Each element of this list represents a list of matrices corresponding to the (optionally multiple) grouping levels of random effects. In each matrix, rows corresponding to missing outcomes have been excluded.
`ylist`	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.
`Slist`	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
`nalist`	a m-dimensional list of group-specific logical vectors, identifying missing outcomes.
`rep`	matrix with m rows where each column identifies the number of repetitions (number of groups) for each grouping level. The first column (outer level) is by definition a vector of 1's.
`k`, `q`, `nall`	number of outcomes, number of random-effects predictors (including the intercept), total number of observations (excluding missing), respectively. While usually all are scalars, in the case of multilevel models `q` can be a numeric vector representing the number of predictors for each level.
`bscov`	a character vector defining the structure of the (co)variance matrix for each level or random effects. See `mixmeta`.
`control`	list of parameters for controlling the fitting process, usually internally set to default values by `mixmeta.control`.
`...`	further arguments passed to or from other methods. Currently not used.

Details

The estimation involves kp fixed-effects coefficients and random-effects parameters, whose number depends on the number of grouping levels and, for each of them, on the chosen `(co)variance structure` for the between-study (co)variance matrices. A maximum of $kq(kq+1)/2$ parameters are needed in the case of or single-level models with unstructured form for the random-effects (co)variance matrix.

(Restricted) maximum likelihood estimators implemented in **mixmeta** rely on two iterative algorithms: `(R)IGLS` and `quasi-Newton` iterative methods. The former implements a (restricted) iterative generalized least squares method, while the latter is based on a Newton-type maximization routine using specific `likelihood functions`. The default estimation method is based on a hybrid procedure, with few runs of of the (R)IGLS algorithm and then quasi-Newton iterations until convergence. This approach is optimal in exploiting the properties of both algorithms, with (R)IGLS being robust to the choice of initial values and quick in getting near the maximum, while the quasi-Newton is fast to converge from that point. Full (R)IGLS or quasi-Newton methods can be alternatively selected using the `control` argument of `mixmeta` (see `mixmeta.control`). Follow the links above for details on each iterative algorithm.

Both estimation algorithms adopt a profiled (or concentrated) approach, where the optimization is expressed only in terms of the random-effects parameters. Cholesky and and QR decompositions

are used for computational stability and efficiency, and for assuring the positive-definiteness of the estimated between-study (co)variance matrix. The method is described in details in Gasparrini and collaborators (2012) (see references below).

Value

These functions return an intermediate list object, with some components then processed and some others added later within `mixmeta.fit` and `mixmeta` to finalize an object of class `"mixmeta"`. See `mixmetaObject`.

Note

As stated earlier, these functions are called internally by `mixmeta.fit`, and are not meant to be used directly. In particular, their code does not contain any check on the arguments provided, which are expected in specific formats. The functions are not exported in the namespace, and only documented for completeness.

The arguments above are prepared by `mixmeta.fit` from its arguments `X`, `Z`, `y`, `S`, `groups`, and `bscov`. The list structure, although requiring more elaborate coding, is computationally more efficient, as it avoids the specification of sparse block-diagonal matrices, especially for meta-analysis involving a large number of studies.

Some parameters of the fitting procedures are determined by the `control` argument, with default set by `mixmeta.control`. No missing values are accepted in the fitting functions. See details on `missing values`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

Pinheiro JC and Bates DM (2000). *Mixed-Effects Models in S and S-PLUS*. New York, Springer Verlag.

Lindstrom MJ and Bates DM (1988). Newton-Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*. **83**(404):1014–1022.

Goldstein H (1986). Multilevel mixed linear model analysis using iterative generalized least squares. *Biometrika*. **73**(1):43.

Goldstein H (1992). Efficient computational procedures for the estimation of parameters in multilevel models based on iterative generalized least squares. *Computational Statistics & Data Analysis*. **13**(1):63–71.

See Also

See mixmeta for the general usage of the functions. See mixmeta.control to determine specific parameters of the fitting procedures. Use the triple colon operator (':::') to access the code of the internal functions, such as glsfit. See mixmeta-package for an overview of the package and modelling framework.

Examples

```
# REML ESTIMATOR: UNIVARIATE MODEL
mod1 <- mixmeta(yC, S=SCC, data=smoking)
summary(mod1)

# ML ESTIMATOR: MULTIVARIATE MODEL
year <- berkey98$pubyear - 1983
mod2 <- mixmeta(cbind(PD,AL) ~ year, S=berkey98[5:7], data=berkey98,method="ml")
print(summary(mod2), digits=3)
round(mod2$Psi,3)

# STRUCTURED BETWEEN-STUDY (CO)VARIANCE
y <- as.matrix(fibrinogen[2:5])
S <- as.matrix(fibrinogen[6:15])
mod3 <- mixmeta(y, S, bscov="hcs")
summary(mod3)

# MULTILEVEL MODEL
mod4 <- mixmeta(effect, var, random= ~ 1|district/study, data=school)
summary(mod4)

# LONGITUDINAL MODEL
mod5 <- mixmeta(logOR~time, S=logORvar, random=~I(time-15)|study, bscov="diag",
  method="ml", data=gliomas)
summary(mod5)
```

---

mixmeta.mm *Method of Moments Estimator for mixmeta Models*

Description

This function implements a method of moments estimator for multivariate and univariate random-effects meta-analysis and meta-regression. It is meant to be used internally and not directly run by the users.

Usage

```
mixmeta.mm(Xlist, ylist, Slist, nalist, k, m, p, nall, control, ...)
```

Arguments

Assuming a meta-analysis or meta-regression based on m independent groups (usually studies) providing a single estimate (unit of analysis), k outcomes and p fixed-effects predictors:

`Xlist`	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
`ylist`	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.
`Slist`	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
`nalist`	a m-dimensional list of group-specific logical vectors, identifying missing outcomes.
`k`, `m`, `p`, `nall`	numeric scalars: number of outcomes, number of groups included in estimation (equal to the length of lists above), number of predictors (including the intercept), total number of observations (excluding missing).
`control`	list of parameters for controlling the fitting process, usually internally set to default values by `mixmeta.control`.
`...`	further arguments passed to or from other methods. Currently not used.

Details

In this method of moments estimator, only a basic random-effects structure is allowed, where each group (usually corresponding to an independent study) provides a single estimate (unit of analysis) for one or multiple outcomes. This implies that the number of groups (*i.e.*, the length of the lists) is identical to the number of units (`m=n`). In addition, only an unstructured form for the(co)variance matrix of the single level of random effects is permitted. Therefore, the estimation involves kp fixed-effects coefficients and $k(k+1)/2$ random-effects parameters, corresponding to the lower triangular entries of the between-study (co)variance matrix.

The method of moment estimator implemented here represents a multivariate extension of the traditional estimator proposed by DerSimonian and Laird (1986), and simplifies to the standard method in the univariate case. The estimator used here is described in Jackson and collaborators (2013) as a generalization of that developed by Chen and collaborators (2012). However, this general version is computationally more intensive, and may turn out to be slow when applied to meta-analysis of a relatively high number of studies. An alternative and computationally faster method of moment estimator was previously proposed by Jackson and collaborators (2010), although it is not invariant to reparameterization. This latter estimator is currently not implemented in **mixmeta**. See references below.

This method of moments estimator is not bounded to provide a positive semi-definite random-effects (co)variance matrix, as shown in the simulation study by Liu and colleagues (2009). Here positive semi-definiteness is forced by setting the negative eigenvalues of the estimated matrix to a positive value close to zero at each iteration (see `control`). Little is known about the impact of such constraint.

Value

This function returns an intermediate list object, with some components then processed and some others added later within `mixmeta.fit` and `mixmeta` to finalize an object of class `"mixmeta"`. See `mixmetaObject`.

Note

As stated earlier, this function is called internally by `mixmeta.fit`, and is not meant to be used directly. In particular, its code does not contain any check on the arguments provided, which are expected in specific formats. The function is however exported in the namespace and documented for completeness.

The arguments above are prepared by `mixmeta.fit` from its arguments `X`, `y` and `S`. The list structure, although requiring more elaborate coding, is computationally more efficient, as it avoids the specification of sparse block-diagonal matrices, especially for meta-analysis involving a large number of studies.

Some parameters of the fitting procedures are determined by the `control` argument, with default set by `mixmeta.control`. No missing values are accepted in the fitting functions. See details on `missing values`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

Jackson D, White IR, Riley RD (2013). A matrix based method of moments for fitting the multivariate random effects model for meta-analysis and meta-regression. *Biometrical Journal*. **55**(2):231–45.

See Also

See `mixmeta` for the general usage of the functions. See `mixmeta.control` to determine specific parameters of the fitting procedures. Use the triple colon operator (':::') to access the code of the internal functions, such as `fbtr`. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# MM ESTIMATOR: UNIVARIATE MODEL
mod1 <- mixmeta(PD ~ pubyear, S=berkey98[,5], data=berkey98, method="mm")
summary(mod1)

# MULTIVARIATE MODEL: REPRODUCE THE RESULTS IN CHEN ET AL. (2012)
```

```
S <- as.matrix(hsls[5:10])
mod2 <- mixmeta(cbind(b1,b2,b3), S, data=hsls, method="mm")
summary(mod2)

# MULTIVARIATE MODEL: REPRODUCE THE RESULTS IN JACKSON ET AL. (2013)
S <- inputcov(hyp[c("sbp_se","dbp_se")], cor=hyp$rho)
mod3 <- mixmeta(cbind(sbp,dbp), S=S, data=hyp, method="mm")
summary(mod3)
```

`mixmeta.vc` *Variance Components Estimator for mixmeta Models*

Description

This function implements a variance components estimator for multivariate and univariate random-effects meta-analysis and meta-regression. It is meant to be used internally and not directly run by the users.

Usage

```
mixmeta.vc(Xlist, ylist, Slist, nalist, k, m, p, nall, control, ...)
```

Arguments

Assuming a meta-analysis or meta-regression based on m independent groups (usually studies) providing a single estimate (unit of analysis), k outcomes and p fixed-effects predictors:

`Xlist`	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
`ylist`	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.
`Slist`	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
`nalist`	a m-dimensional list of group-specific logical vectors, identifying missing outcomes.
`k`, `m`, `p`, `nall`	numeric scalars: number of outcomes, number of groups included in estimation (equal to the length of lists above), number of predictors (including the intercept), total number of observations (excluding missing).
`control`	list of parameters for controlling the fitting process, usually internally set to default values by `mixmeta.control`.
`...`	further arguments passed to or from other methods. Currently not used.

Details

In this variance components estimator, only a basic random-effects structure is allowed, where each group (usually corresponding to an independent study) provides a single estimate (unit of analysis) for one or multiple outcomes. This implies that the number of groups (*i.e.*, the length of the lists) is identical to the number of units (`m=n`). In addition, only an unstructured form for the(co)variance matrix of the single level of random effects is permitted. Therefore, the estimation involves kp fixed-effects coefficients and $k(k+1)/2$ random-effects parameters, corresponding to the lower triangular entries of the between-study (co)variance matrix.

The procedure is based on the estimate of the between-group (co)variance as the difference between the marginal (co)variance and the average within-group (co)variance. This in turn requires the estimate of the marginal (co)variance, obtained by the residuals of the fitted model. The procedure is iterative, with the current estimate of the between-group (co)variance plugged into a generalized least square (GLS) routine. Starting values are provided by a fixed-effects estimator (see `mixmeta.fixed`). The algorithm is fast and generally converges with few iterations.

Similar versions of this estimator has been previously proposed. Berkey and collaborators (1998) simply called it GLS method, and a non-iterative approach was proposed by Ritz and collaborators (2008), referred to as MVEE3 in their article. A non-iterative version for univariate models is discussed in Sidik and Jonkman (2007). The results from Berkey and collaborators (1998) are reproduced in the example below.

In the original approach, the estimate of the marginal (co)variance is obtained from the sum of the residual components using a denominator equal to `m-p`. Following the development proposed by Kauermann and Carroll (2001) and Fay and Graubard (2001) in the context of sandwich (co)variance estimators, then discussed by Lu and collaborators (2007), an adjusted denominator can be computed as a quantity derived from the hat matrix. This method is expected to perform better in the presence of missing values and small data sets. This alternative adjustment is chosen by default by setting `vc.adj=TRUE` in the `control` argument.

This variance component estimator is not bounded to provide a positive semi-definite between-study (co)variance matrix, as shown in the simulation study by Liu and colleagues (2009). Here positive semi-definiteness is forced by setting the negative eigenvalues of the estimated matrix to a value close to zero at each iteration (see `control`). Little is known about the impact of such constraint.

Value

This function returns an intermediate list object, with some components then processed and some others added later within `mixmeta.fit` and `mixmeta` to finalize an object of class `"mixmeta"`. See `mixmetaObject`.

Note

As stated earlier, this function is called internally by `mixmeta.fit`, and is not meant to be used directly. In particular, its code does not contain any check on the arguments provided, which are expected in specific formats. The function is however exported in the namespace and documented for completeness.

The arguments above are prepared by `mixmeta.fit` from its arguments `X`, `y` and `S`. The list structure, although requiring more elaborate coding, is computationally more efficient, as it avoids the specification of sparse block-diagonal matrices, especially for meta-analysis involving a large number of studies.

Some parameters of the fitting procedures are determined by the `control` argument, with default set by `mixmeta.control`. No missing values are accepted in the fitting functions. See details on `missing values`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Gasparrini A, Armstrong B, Kenward MG (2012). Multivariate meta-analysis for non-linear and other multi-parameter associations. *Statistics in Medicine*. **31**(29):3821–3839. [Freely available **here**].

Ritz J, Demidenko E, Spiegelman G (2008). Multivariate meta-analysis for data consortia, individual patient meta-analysis, and pooling projects. *Journal of Statistical Planning and Inference*. **139**(7):1919–1933.

Berkey CS, et al. (1998). Meta-analysis of multiple outcomes by regression with random effects. *Statistics in Medicine*. **17**(22):2537–2550.

Liu Q, et al (2009). A two-stage hierarchical regression model for meta-analysis of epidemiologic nonlinear dose-response data. *Computational Statistics and Data Analysis*. **53**(12):4157–4167

Sidik K, Jonkman JN (2007). A comparison of heterogeneity variance estimators in combining results of studies. *Statistics in Medicine*. **26**(9):1964–81.

See Also

See `mixmeta` for the general usage of the functions. See `mixmeta.control` to determine specific parameters of the fitting procedures. Use the triple colon operator (':::') to access the code of the internal functions, such as `sumlist`. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# VC ESTIMATOR: UNIVARIATE MODEL
mod1 <- mixmeta(PD ~ pubyear, S=berkey98[,5], data=berkey98, method="vc")
summary(mod1)

# VC ESTIMATOR: MULTIVARIATE MODEL
mod2 <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98,
  method="vc")
summary(mod2)

# VC ESTIMATOR: NON-ITERATIVE VERSION
mod3 <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98,
  method="vc", control=list(maxiter=1))
summary(mod3)

# VARIANCE COMPONENTS ESTIMATOR: REPLICATE THE RESULTS IN BERKEY ET AL. (1998)
```

```
mod4 <- mixmeta(cbind(PD,AL) ~ I(pubyear-1983), S=berkey98[5:7], data=berkey98,
  method="vc", control=list(vc.adj=FALSE))
summary(mod4)
```

`mixmetaCovStruct` *(Co)variance Structures for mixmeta Models*

Description

Alternative options for the (co)variance structure of the random effects random effects in meta-analytical models, usually defined through the argument `bscov` of the function `mixmeta`.

Options

Assuming a meta-analysis or meta-regression based on k outcomes, for each grouping level with q random-effects predictors the matrix can be specified in various forms listed below. For multivariate models with multiple predictors, the order implies a sequence of q parameters for each k outcomes. These are the options:

- `unstr`: an unstructured form for a general positive-definite matrix. The matrix is represented by $kq(kq+1)/2$ unrestricted parameters defined as the upper triangular entries of its Cholesky decomposition.
- `diag`: a diagonal positive-definite matrix. The matrix is represented by kq unrestricted parameters defined as the logarithm of the diagonal values.
- `id`: a multiple of the identity positive-definite matrix. The matrix is represented by a single unrestricted parameter defined as the logarithm of the diagonal value.
- `cs`: a positive-definite matrix with compound symmetry structure. The matrix is represented by 2 unrestricted parameters defined as the logarithm of the identical diagonal value and the transformed correlation. The latter is parameterized so to obtain a correlation value between $-1/(kq-1)$ and 1, in order to ensure positive-definiteness.
- `hcs`: a positive-definite matrix with heterogeneous compound symmetry structure. The matrix is represented by $kq+1$ unrestricted parameters defined as the logarithm of the diagonal values and the transformed correlation. The latter is parameterized so to obtain a correlation value between $-1/(kq-1)$ and 1, in order to ensure positive-definiteness.
- `ar1`: a positive-definite matrix with autoregressive structure of first order. The matrix is represented by 2 unrestricted parameters defined as the logarithm of the identical diagonal value and the logistic transformed correlation. The latter is parameterized so to obtain a correlation value between -1 and 1.
- `har1`: a positive-definite matrix with heterogeneous autoregressive structure of first order. The matrix is represented by $kq+1$ unrestricted parameters defined as the logarithm of the diagonal value and the logistic transformed correlation. The latter is parameterized so to obtain a correlation value between -1 and 1.
- `prop`: a positive-definite matrix proportional to that provided by the user through the argument `Psifix` in the control list (see `mixmeta.control`). The matrix is represented by 1 unrestricted parameter defined as the logarithm of the multiplier.

- `cor`: a positive-definite matrix with correlation structure provided by the user through the argument `Psifix` (with `cov2cor`) in the control list (see `mixmeta.control`). The matrix is represented by k unrestricted parameters defined as the logarithm of the diagonal values.
- `fixed`: a known matrix provided by the user through the argument `Psifix` in the control list (see `mixmeta.control`). The matrix is known and no parameters are needed to represent it.

Details

Structures other than `unstr` are only available for models estimated through (restricted) maximum likelihood.

The unrestricted parameters defining the random-effects (co)variance matrix (or matrices for multilevel models) are estimated in the iterative optimization algorithm (see `mixmeta.ml`). Although rarely needed and not recommeded, the user can provided a starting value of the (co)variance matrix, from which the parameters are derived (see `mixmeta.control`).

Note

The choice of structures can affect the performance of the optimization procedure, determining forms of likelihood surfaces which induce convergence to local maxima. In particular, structures such as multiple of identity or proportional to a fixed matrix are based on strong assumptions and should be used with caution.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Pinheiro JC and Bates DM (2000). *Mixed-Effects Models in S and S-PLUS*. New York, Springer Verlag.

See Also

See `mixmeta`. See `lm` or `glm` for standard regression functions. See `mixmeta-package` for an overview of this modelling framework.

Examples

```
# UNSTRUCTURED AND STRUCTURED BETWEEN-STUDY (CO)VARIANCE
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
mod1 <- mixmeta(y, S)
summary(mod1)
mod1$Psi

# DIAGONAL
mod2 <- mixmeta(y, S, bscov="diag")
summary(mod2)
```

```
mod2$Psi

# HETEROGENEOUS COMPOUND SYMMETRY
mod3 <- mixmeta(y, S, bscov="hcs")
summary(mod3)
mod3$Psi

# PROPORTIONAL
mod4 <- mixmeta(y, S, bscov="prop", control=list(Psifix=diag(3)+1))
summary(mod4)
mod4$Psi

# CORRELATION
Psicor <- matrix(0.2, 3, 3) ; diag(Psicor) <- 1
mod5 <- mixmeta(y, S, bscov="cor", control=list(Psifix=Psicor))
summary(mod5)
mod5$Psi
```

`mixmetaFormula` *Formulae in mixmeta Models*

Description

Meta-analytical models fitted with `mixmeta` are defined by specific formulae in its arguments `formula` and `random`. The formulae offer compact symbolic expressions with form `y ~ x + z`, where the response `y` in the left-hand side of the operator `~` is modelled in terms of meta-predictors `x` and `z` in the right-hand side. Terms are separated by `+`, and additional syntactic operators and existing functions can be used within a formula to specify transformations such as categorization and interactions, among others, as in standard formula expressions (see `formula` for details). The usage of formulae in **mixmeta** for the random-effects part follows closely the definition in the the **nlme** package.

Fixed-effects formula

The argument `formula` of `mixmeta` defines the fixed-effects part. Models for meta-analysis with no meta-predictors can be specified using the form `y ~ 1`, or alternatively including only the term `y` (in this case, the formula is reconstructed internally). Multivariate models can be defined by using a matrix-type `y`, with columns as multiple outcomes, or directly in the formula with form `cbind(y1 + y2) ~ 1`. In meta-regression models, other terms are added in the right-hand side of the formula as `y ~ x1 + ... + xp`, defining the linear meta-predictor. In multivariate meta-regression, the same linear predictor is specified for each outcome.

Random-effects formula or formulae

The argument `random` of `mixmeta` defines the random-effects part. When this is not specified, it is assumed that each row of data is from an independent study and assigned to a different group, as in standard meta-analytical models. If provided, this is usually represented by a one-sided formula whose basic random-intercept form is `~ 1 | g`. The term `g` at the right-hand side of the special operator `|` is the grouping factor, always required in a single random-effects formula. A more

complex random-effects part can be also specified by `~ z1 + ... + zq | g`, where the terms in the left-hand side defines a linear meta-predictor, with syntax identical to the usual formulae.

The argument `random` also accepts a list of one-sided formulae referring to multiple random-effects levels in multilevel meta-analytical models. In this case, levels are assumed to be nested in the order of the list, from the lowest (outer) to the highest (inner), consistently with the grouping factors. These are usually defined by different terms in the right-hand side of the `|` operator, although in the list form can also be provided as names of the list components. This information is used internally to reconstruct the grouping structure and the random-effects design matrices. Each level can have different linear predictors, but if these are identical across levels the random-effects part can be defined by a single equation `~ z | g1 / g2`, where the special operator `/` separates the grouping factors `g2` nested in `g1`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See `mixmeta`. See `formula` for standard regression formulae. See `mixmeta-package` for an overview of this modelling framework.

Examples

```
# STANDARD RANDOM-EFFECTS META-ANALYSIS (WITH DIFFERENT SYNTAXES)
mixmeta(logor, logorvar, data=bcg)
mixmeta(logor ~ 1, logorvar, data=bcg)

# META-REGRESSION
mixmeta(logor ~ ablat, logorvar, data=bcg)

# MULTIVARIATE MODEL
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98)

# NON-STANDARD MODEL: REPEATED MEASURED WITHING THE SAME GROUPS
mixmeta(effect, var, random= ~ 1|district, data=school)
mixmeta(absrisk, var, random= ~ 1|trial, data=thrombolytic)

# NON-STANDARD MODEL: MORE COMPLEX RANDOM-EFFECTS PREDICTOR
mixmeta(logOR~time, logORvar, random= ~ I(time-15)|study, data=gliomas)

# MULTILEVEL MODEL (WITH DIFFERENT SYNTAXES)
mixmeta(effect, var, random= ~ 1|district/study, data=school)
mixmeta(effect, var, random=list(~ 1|district, ~ 1|study), data=school)
mixmeta(effect, var, random=list(district = ~ 1, study = ~ 1), data=school)
```

`mixmetaObject` *mixmeta Objects*

Description

An object returned by the `mixmeta` function, inheriting from class `"mixmeta"`, and representing a fitted univariate or multivariate meta-analytical model.

Value

Objects of class `"mixmeta"` are lists with defined components. Dimensions of such components may refer to k outcome parameters, p fixed-effects and q random-effects predictors, m groups and n units used for fitting the model (the latter can be different from those originally selected due to missing). Depending on the type of meta-analytical model, the following components can bu included in a legitimate `mixmeta` object:

`coefficients` a kp-dimensional vector of the fixed-effects coefficients.

`vcov` estimated $kp \times kp$ (co)variance matrix of the fixed-effects coefficients.

`Psi` the estimated $kq \times kq$ random-effects (co)variance matrix, or a list of matrices for multilevel models. Only for random-effects models.

`residuals` a n-dimensional vector (for univariate models) or $n \times k$ matrix (for multivariate models) of residuals, that is observed minus fitted values.

`fitted.values` a n-dimensional vector (for univariate models) or $n \times k$ matrix (for multivariate models) of fitted mean values.

`df.residual` the residual degrees of freedom.

`rank` the numeric rank of the fixed-effects part of the fitted model.

`logLik` the (restricted) log-likelihood of the fitted model. Set to `NA` for non-likelihood models.

`converged`, `niter` for models with iterative estimation methods, logical scalar indicating if the algorithm eventually converged and number or iterations, respectively.

`par` parameters estimated in the optimization process when using likelihood-based estimators. These correspond to trasformations of entries of the random-effects (co)variance matrix, dependent on chosen `(co)variance structure`. For multilevel models, the vector includes the parameters of multiple matrices. Returned also for full (R)IGLS optimization, even if not directly used. See also the `mixmeta.ml` for details.

`hessian` Hessian matrix of the estimated parameters in `par` above, only returned if `hessian=TRUE` in `mixmeta.control`. See the related `optimizations algorithms` for details.

`dim` list with the following components: `k` (scalar, number of outcome parameters), `n` (scalar, number of units included in estimation, which could be lower than the total number in the presence of missing values), `m` (scalar, number of outer-level groups), `p` (scalar, number of fixed-effects predictors), `q` (scalar or vector,number of random-effects predictors).

df	list with the following scalar components: nall (number of observations used for estimation, excluding missing values), nobs (equal to nall, minus the number of fixed-effects coefficients in REML models), fixed (number of estimated fixed-effects coefficients), random (number of estimated random-effects (co)variance terms).
lab	list with the following label vectors: k for the outcome parameters, and p and q for the fixed and random-effects predictors, respectively (including intercept). The last one can be a list for multilevel models.
S	a $n \times k(k+1)/2$ matrix, where each row represents the vectorized entries of the lower triangle of the related within-unit (co)variance error matrix, taken by column. See mixmeta.
call	the function call.
formula	the formula for the fixed-effects part of the model. See mixmetaFormula.
model	the model frame used for fitting. Reported if model=TRUE in mixmeta. See model.frame.
terms	the terms object representing the fixed-effects part of the fitted model.
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting.
na.action	(where relevant) information returned by model.frame on the special handling of NAs. See info on missing values.
method	the estimation method.
random	the formula (or list of formulae for multilevel models) for the random-effects part of the model. See mixmetaFormula.
bscov	a string defining the random-effects (co)variance structure in likelihood based models. See model.frame.mixmeta.
control	a list with the values of the control arguments used, as returned by mixmeta.control.

Methods

A number of methods functions are available for mixmeta objects, most of them common to other regression functions.

Specifically-written method functions are defined for predict (standard predictions) and blup (best linear unbiased predictions). The method function simulate produces simulated outcomes from a fitted model, while qtest performs the Cochran Q test for heterogeneity. Other methods have been produced for summary, logLik, coef, and vcov.

Specific methods are also available for model.frame and model.matrix. In particular, the former produces the model frame (a data frame with special attributes storing the variables used for fitting) with the additional class "data.frame.mixmeta". A method terms is also available for extracting the terms object (only for fixed-effects or full). Methods na.omit and na.exclude for this class are useful for the handling of missing values in mixmeta objects.

Printing functions for the objects of classes defined above are also provided. anova methods for performing tests in mixmeta objects are in development.

All the methods above are visible (exported from the namespace) and documented. In additions, several default method functions for regression are also applicable to objects of class "mixmeta", such as fitted, residuals, AIC and BIC, drop1 and add1, or update, among others.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See `mixmeta`. See `lm` or `glm` for standard regression functions. See `mixmeta-package` for an overview of this modelling framework.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL)~pubyear, S=berkey98[5:7], data=berkey98)

# INSPECT THE OBJECT
names(model)

# LABELS
model$lab

# FORMULA
model$formula

# CONVERGED?
model$converged
```

---

mixmetaSim *Simulating Responses for mixmeta Models*

Description

These functions simulate sets of responses (either univariate or multivariate) for a group of units, in terms of their mean (expected) values and within and between-group (co)variances. These sets of outcomes can be used in meta-analytical models for simulation purposes.

Usage

```
mixmetaSim(y, S, Psi, random, data, nsim=1, seed=NULL, ...)

## S3 method for class 'mixmeta'
simulate(object, nsim=1, seed=NULL, ...)
```

Arguments

In order to simulate k outcomes for n units:

y	a n-dimensional vector (for simulating univariate responses) or $m \times k$ matrix (for simulating multivariate responses) of mean (expected) outcomes.
S	series of within-unit variances (or (co)variance matrices for multivariate models) of the estimated outcome(s). For the list of accepted format, see the argument with the same name in mixmeta. Covariances or more complex error structures can be passed through additional arguments. See Details below.
Psi	the random-effects (co)variance matrix (or a list of matrices for multilevel models) of the outcomes. Dimension must be consistent with the specification of the random-effects structure in random.
random	a one-sided formula (or a list of formulae for multilevel models) offering a symbolic description of the linear predictor(s) and grouping structure for the random-effects part of the model. See the argument with the same name in mixmeta.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame), optionally containing the variables in y, S, and random.
nsim	number of simulation sets.
seed	an object specifying if and how the random number generator should be initialized.
object	an object of class "mixmeta".
...	further optional arguments.

Details

The set(s) of responses can be simulated either from a fitted model, using the method function simulate for objects of class "mixmeta", or directly through the function mixmetaSim. In the former case, the fitted values from the model are used as mean (expected) outcomes, together with the within-unit and estimated random-effects (co)variance structure. In the latter option, this information need to be provided by the user in the correct dimensions and forms.

Additional arguments can be passed in '...'. Specifically, arguments Scor and addSlist can be added to input missing within-unit error covariances, or to specify more complex within-unit error structures, respectively. Another argument can be checkPD (logical), that checks the semi-positive definiteness of the matrices). See mixmeta.control for details.

The functions simulate the responses for each study separately from a marginal multivariate normal distribution with mean equal to the expected values and (co)variance equal to the sum of the within-unit errors and random-effects components. The computation is identical to that implemented in the function mvrnorm of the package **MASS**, involving a eigen decomposition of the marginal (co)variance matrix. Numerical negative definiteness is checked, and positive semi-definiteness then forced by truncating the eigenvalues at a value close to zero (see control).

Value

If nsim=1, a matrix or vector of simulated k outcomes for the n units. If more simulation sets are required (nsim higher than 1), a list of matrices or vectors.

Note

Studies with missing values in the fitted values or in the components of the within (co)variances are excluded by `simulate`. Missing values are instead not accepted in `metaSim`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See `simulate` for the general method function. See `inputcov` for inputting correlations. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# RUN A MODEL
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98)

# SIMULATE A NEW SET OF OUTCOMES
simulate(model)

# SIMULATE FROM SCRATCH: 3 OUTCOMES, 8 STUDIES
(y <- matrix(0, 8, 3))
(S <- inputcov(matrix(runif(8*3, 0.1, 2), 8, 3, dimnames=list(NULL,
  c("V1","V2","V3"))), cor=c(0,0.5,0.7)))
(Psi <- inputcov(1:3, cor=0.3))
mixmetaSim(y, S, Psi)

# 2 SIMULATION SETS
mixmetaSim(y, S, Psi, nsim=2)
```

---

ml.igls *IGLS and RIGLS Iterative Algorithms for mixmeta Models*

Description

These functions implements (restricted) iterative generalized least squares (IGLS and RIGLS) algorithms for (restricted) maximum likelihood estimators for random-effects meta-analytical models. They are meant to be used internally and not directly run by the users.

Usage

```
ml.igls(Psi, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const, bscov,
  fix, control)

reml.rigls(Psi, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const, bscov,
  fix, control)

igls.iter(Psi, Qlist, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, bscov,
  fix, control)

rigls.iter(Psi, Qlist, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, bscov,
  fix,  control)
```

Arguments

Assuming a meta-analysis or meta-regression based on n units aggregated within m (outer-level) groups, k outcomes, p fixed-effects predictors, and q random-effects predictors:

`Psi`	a matrix (or a list of matrices for multilevel models) representing the initial estimate of the random-effects (co)variance matrix.
`Xlist`	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
`Zlist`	a m-dimensional list of group-specific design matrices for the random-effects part of the model. Each element of this list represents a list of matrices corresponding to the (optionally multiple) grouping levels of random effects. In each matrix, rows corresponding to missing outcomes have been excluded.
`Qlist`	a m-dimensional list of group-specific design matrices mapping the random-effects parameters to be estimated in `Psi`. See references below for details.
`ylist`	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.
`Slist`	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
`nalist`	a m-dimensional list of group-specific logical vectors, identifying missing outcomes.
`rep`	matrix with m rows where each column identifies the number of repetitions (number of groups) for each grouping level. The first column (outer level) is by definition a vector of 1's.
`k`, `q`, `nall`	number of outcomes, number of random-effects predictors (including the intercept), total number of observations (excluding missing), respectively. While usually all are scalars, in the case of multilevel models `q` can be a numeric vector representing the number of predictors for each level.
`const`	value of the constant to be included in the (restricted) likelihood, therefore not computed in the iterative algorithms.
`bscov`	a character vector defining the structure of the (co)variance matrix for each level or random effects. See `mixmeta`.

fix	a matrix (or optionally a list of matrices for multilevel models) defining the fixed components of the random-effects part of the model. See mixmeta.control for details.
control	list of parameters for controlling the fitting process, usually internally set to default values by mixmeta.control.

Details

These functions are called internally by the fitting functions mixmeta.ml and mixmeta.reml to perform (R)IGLS optimization algorithms for estimating random-effects meta-analytical models.

These estimators are not sensitive to the choice of the starting values, and quickly converge to the vicinity of the (restricted) maximum likelihood. The starting values in Psi are therefore defined by default as a matrix (or matrices) with a diagonal form and 0.001 variances, or otherwise selected by the user in the control argument of mixmeta (see mixmeta.control).

The functions ml.igls and reml.rigls first produce a design matrix that maps the entries of Psi, and then call iter.igls and iter.rigls, respectively, to obtain updated results at each iteration following a (R)IGLS procedure described in Goldstein and colleagues (1992). Convergence is assessed as (lack of) changes in Psi. Positive semi-definiteness is forced by setting the negative eigenvalues of the estimated matrix to a value close to 0 at each iteration (see control).

Value

The functions ml.igls and reml.rigls return an intermediate list object, with components corresponding to the estimated random-effects (co)variance matrix (or list of matrices), its parameters, the maximum (restricted) log-likelihood value, an indicator of convergence, and the number of iterations. These are then re-processed, with other components added later within other functions to finalize an object of class "mixmeta" (see mixmetaObject). The functions iter.igls and iter.rigls return an updated version of Psi.

Note

As stated earlier, these functions are called internally by mixmeta.ml and mixmeta.reml, and are not meant to be used directly. In particular, their code does not contain any check on the arguments provided, which are expected in specific formats. They are however exported in the namespace and documented for completeness.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Goldstein H (1992). Efficient computational procedures for the estimation of parameters in multilevel models based on iterative generalized least squares. *Computational Statistics & Data Analysis*. **13**(1):63–71.

Goldstein H (1986). Multilevel mixed linear model analysis using iterative generalized least squares. *Biometrika*. **73**(1):43–56.

Goldstein H (1989). Restricted unbiased iterative generalized least-squares estimation. *Biometrika*. **76**(3):622–623.

See Also

See `mixmeta.fit` and `mixmeta.ml` for additional info on the fitting procedures. See `mixmeta.control` to determine specific parameters of the fitting procedures. See `mixmetaCovStruct` for (co)variance structures. See `mixmeta-package` for an overview of the package and modelling framework.

`ml.loglik.fn` *Likelihood Functions for mixmeta Models*

Description

These functions compute the value of the log-likelihood and the related vectors of first partial derivatives for random-effects meta-analytical models, in terms of model parameters. They are meant to be used internally and not directly run by the users.

Usage

```
ml.loglik.fn(par, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const,
  bscov, fix)
ml.loglik.gr(par, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const,
  bscov, fix)

reml.loglik.fn(par, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const,
  bscov, fix)
reml.loglik.gr(par, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const,
  bscov, fix)
```

Arguments

Assuming a meta-analysis or meta-regression based on n units aggregated within m (outer-level) groups, k outcomes, p fixed-effects predictors, and q random-effects predictors:

`par`	a vector representing the random-effects parameters defining the random-effects (co)variance matrix (or multiple matrices for multilevel models).
`Xlist`	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
`Zlist`	a m-dimensional list of group-specific design matrices for the random-effects part of the model. Each element of this list represents a list of matrices corresponding to the (optionally multiple) grouping levels of random effects. In each matrix, rows corresponding to missing outcomes have been excluded.
`ylist`	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.

Slist	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
nalist	a m-dimensional list of group-specific logical vectors, identifying missing outcomes.
rep	matrix with m rows where each column identifies the number of repetitions (number of groups) for each grouping level. The first column (outer level) is by definition a vector of 1's.
k, q, nall	number of outcomes, number of random-effects predictors (including the intercept), total number of observations (excluding missing), respectively. While usually all are scalars, in the case of multilevel models q can be a numeric vector representing the number of predictors for each level.
const	value of the constant to be included in the (restricted) likelihood, therefore not computed in the iterative algorithms.
bscov	a character vector defining the structure of the (co)variance matrix for each level or random effects. See mixmeta.
fix	a matrix (or optionally a list of matrices for multilevel models) defining the fixed components of the random-effects part of the model. See mixmeta.control for details.

Details

These functions are called internally by fitting functions, in particular ml.newton and reml.newton, to compute the (restricted) log-likelihood and its first partial derivatives in terms of random-effects parameters for meta-analytical models.

These functions actually specify the *profiled* version of the (restricted) likelihood, expressed only in terms of random-effects parameters, while the estimate of the fixed-effects coefficients is computed at each iteration using a generalized least squares estimator, based on the current value of the between-study (co)variance matrix. At convergence, the value of this profiled version is identical to the full (restricted) likelihood. This approach is computationally efficient, as it reduces the number of parameters in the optimization routine, especially for meta-regression models.

The random-effects parameters in par depends on the chosen structure(s) for the random-effects (co)variance matrix (or multiple matrices for multilevel models). The parameterization ensures positive-definiteness. A Cholesky decomposition is then performed on the marginal (co)variance matrix in order to re-express the problem as standard least square equations, an approach which speeds up the computation of matrix inverses and determinants. These equations are finally solved through a QR decomposition, which guarantees stability. More details are provided in the references below.

Some parameters of the fitting procedures are determined through mixmeta.control. Specifically, the user can obtain the Hessian matrix of the estimated parameters (appropriately transformed, see mixmetaCovStruct) in the optimization function by setting hessian=TRUE, and specific settings of the optimization process can be defined by the control list argument optim. These values are passed to the optimization function optim.

Value

`ml.loglik.fn` and `reml.loglik.fn` return the value of the (restricted) log-likelihood for a given set of parameters in `par`. `ml.loglik.gr` and `reml.loglik.gr` return instead the related vector of first partial derivatives.

Note

As stated earlier, these functions are called internally by `mixmeta.ml` and `mixmeta.reml`, and are not meant to be used directly. In particular, their code does not contain any check on the arguments provided, which are expected in specific formats. They are however exported in the namespace and documented for completeness.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Lindstrom MJ and Bates DM (1988). Newton-Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*. **83**(404):1014–1022.

Harville DA (1977) Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*. **72**(358):320–338.

Pinheiro JC and Bates DM (2000). *Mixed-Effects Models in S and S-PLUS*. New York, Springer Verlag.

See Also

See `mixmeta.fit` and `mixmeta.ml` for additional info on the fitting procedures. See `mixmeta.control` to determine specific parameters of the fitting procedures. See `mixmetaCovStruct` for (co)variance structures. See `chol` and `qr` for info on the Cholesky and QR decomposition. See `mixmeta-package` for an overview of the package and modelling framework.

`ml.newton` *Quasi-Newton Iterative Algorithms for mixmeta Models*

Description

These functions implement quasi-Newton iterative algorithms for (restricted) maximum likelihood estimators for random-effects meta-analytical models. They are meant to be used internally and not directly run by the users.

Usage

```
ml.newton(Psi, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const,
  bscov, fix, control)

reml.newton(Psi, Xlist, Zlist, ylist, Slist, nalist, rep, k, q, nall, const,
  bscov, fix, control)
```

Arguments

Assuming a meta-analysis or meta-regression based on n units aggregated within m (outer-level) groups, k outcomes, p fixed-effects predictors, and q random-effects predictors:

`Psi`	a matrix (or a list of matrices for multilevel models) representing the initial estimate of the random-effects (co)variance matrix.
`Xlist`	a m-dimensional list of group-specific design matrices for the fixed-effects part of the model. Rows corresponding to missing outcomes have been excluded.
`Zlist`	a m-dimensional list of group-specific design matrices for the random-effects part of the model. Each element of this list represents a list of matrices corresponding to the (optionally multiple) grouping levels of random effects. In each matrix, rows corresponding to missing outcomes have been excluded.
`ylist`	a m-dimensional list of group-specific vectors of estimated outcomes. Entries corresponding to missing outcomes have been excluded.
`Slist`	a m-dimensional list of within-group (co)variance matrices of estimated outcomes. Rows and columns corresponding to missing outcomes have been excluded.
`nalist`	a m-dimensional list of group-specific logical vectors, identifying missing outcomes.
`rep`	matrix with m rows where each column identifies the number of repetitions (number of groups) for each grouping level. The first column (outer level) is by definition a vector of 1's.
`k`, `q`, `nall`	number of outcomes, number of random-effects predictors (including the intercept), total number of observations (excluding missing), respectively. While usually all are scalars, in the case of multilevel models `q` can be a numeric vector representing the number of predictors for each level.
`const`	value of the constant to be included in the (restricted) likelihood, therefore not computed in the iterative algorithms.
`bscov`	a character vector defining the structure of the (co)variance matrix for each level or random effects. See `mixmeta`.
`fix`	a matrix (or optionally a list of matrices for multilevel models) defining the fixed components of the random-effects part of the model. See `mixmeta.control` for details.
`control`	list of parameters for controlling the fitting process, usually internally set to default values by `mixmeta.control`.

Details

These functions are called internally by the fitting functions `mixmeta.ml` and `mixmeta.reml` to perform quasi-Newton iterative optimization algorithms for estimating random-effects meta-analytical models.

Starting values for the iterations are defined by `Psi`, representing a random-effects (co)variance matrix (or a list of matrices for multilevel models). In the default hybrid procedure (see `mixmeta.ml`), these are provided using few iterations of a `(R)IGLS` algorithm. If a full quasi-Newton method is used, the starting values are instead defined by default as a matrix (or matrices) with a diagonal form and 0.001 variances, or otherwise selected by the user in the `control` argument of `mixmeta` (see `mixmeta.control`).

The functions first re-define `Psi` as a set of random-effects parameters, depending on the chosen `structure(s)`, using parameterizations that ensure the positive-definiteness of the estimated matrix (or matrices). Then, the function `optim` with `method="BFGS"` is called internally to perform the quasi-Newton optimization, using specific `likelihood functions` that compute the value of the (restricted) likelihood and (optionally) the vector of its first partial derivatives. The latter are used only in the case of basic random-effects structures, or otherwise the derivatives are computed numerically.

Some parameters of the optimization procedures are determined through `mixmeta.control`. Specifically, the user can obtain the Hessian matrix of the estimated parameters (appropriately transformed, see `mixmetaCovStruct`) in the optimization function by setting `hessian=TRUE`, and specific settings of the optimization process can be defined by the control list argument `optim`. These values are passed to the optimization function `optim`.

Value

These functions return an intermediate list object, with components corresponding to the estimated random-effects (co)variance matrix (or list of matrices), the maximum (restricted) log-likelihood value, an indicator of convergence, the number of iterations, and optionally the Hessian matrix. These are then re-processed, with other components added later within other functions to finalize an object of class `"mixmeta"`. See `mixmetaObject`.

Note

As stated earlier, these functions are called internally by `mixmeta.ml` and `mixmeta.reml`, and are not meant to be used directly. In particular, their code does not contain any check on the arguments provided, which are expected in specific formats. They are however exported in the namespace and documented for completeness.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Lindstrom MJ and Bates DM (1988). Newton-Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*. **83**(404):1014–1022.

Harville DA (1977) Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*. **72**(358):320–338.

Pinheiro JC and Bates DM (2000). *Mixed-Effects Models in S and S-PLUS*. New York, Springer Verlag.

See Also

See `mixmeta.fit` and `mixmeta.ml` for additional info on the fitting procedures. See `mixmeta.control` to determine specific parameters of the fitting procedures. See `mixmetaCovStruct` for (co)variance structures. See `chol` and `qr` for info on the Cholesky and QR decomposition. See `mixmeta-package` for an overview of the package and modelling framework.

`model.frame.mixmeta` *Extract Model Frame and Design Matrix from mixmeta Objects*

Description

These method functions return the model frame and design matrix for meta-analytical models represented in objects of class `"mixmeta"`.

Usage

```
## S3 method for class 'mixmeta'
model.frame(formula, ...)

## S3 method for class 'mixmeta'
model.matrix(object, ...)
```

Arguments

`object`, `formula`	an object of class `"mixmeta"`.
`...`	further arguments passed to or from other methods.

Details

The model frame is produced by `mixmeta` when fitting the meta-analytical model, and stored in the `mixmeta` object if argument `model=TRUE`. Alternatively, the model frame is directly returned from a call to `mixmeta` with argument `method="model.frame"`. The method function `model.frame` simply extracts the saved model frame if available, or otherwise evaluates a call to `mixmeta` when `method="model.frame"`.

The method function `model.matrix` extracts the design matrix for the fixed-effects part of a fitted meta-analytical model. It first extract the model frame by calling `model.frame`, and then passes the call to the default method.

Note that the model frame of mixmeta models consist of terms for both the fixed and random-effects parts, the latter including also the grouping factors. This information can be used to reconstruct the proper model frame or matrix for each part.

These methods functions are similar to those provided for regression objects lm and lm.

Value

For model.frame, a data.frame with special attributes (see the default method model.frame) and the additional class "data.frame.mixmeta".

For model.matrix, the design matrix used to fit the model.

Note

The reason why these specific method functions are made available for class mixmeta, and in particular why a new class "data.frame.mixmeta" has been defined for model frames, lies in the special handling of missing values in multivariate meta-analysis models fitted with mixmeta. Methods na.omit and na.exclude for class "data.frame.mixmeta" are useful for properly accounting for missing values when fitting these models.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See the default methods model.frame and model.matrix. See na.omit and na.exclude on the handling of missing values. See mixmeta-package for an overview of the package and modelling framework.

Examples

```
# RUN THE MODEL AND SUMMARIZE THE RESULTS
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98,
  method="ml")

# MODEL FRAME
model$model
model.frame(model)
update(model, method="model.frame")
class(model.frame(model))

# MODEL MATRIX
model.matrix(model)
```

`na.omit.data.frame.mixmeta` *Handling Missing Values in mixmeta Models*

Description

These method functions exclude rows corresponding to units with invalid missing pattern from model frames of class `"data.frame.mixmeta"`. This guarantees the correct handling of missing values while fitting meta-analytical models.

Usage

```
## S3 method for class 'data.frame.mixmeta'
na.omit(object, ...)

## S3 method for class 'data.frame.mixmeta'
na.exclude(object, ...)
```

Arguments

`object`	an object of class `"data.frame.mixmeta"`.
`...`	further arguments passed to or from other methods.

Details

A model frame of class `"data.frame.mixmeta"` is produced by `mixmeta`. A call to `na.omit` or `na.exclude` removes from the model frame the rows corresponding to studies with invalid missing pattern. In addition, a `na.action` attribute is added to the model frame, namely a numeric vector corresponding to the removed rows and class `"omit"` or `"exclude"`, respectively. This information is used by `naresid` and `napredict` to deal with missing values in functions such as `fitted`, `residuals`, `predict` and `blup`, among others.

The definition of missing, identifying an invalid missing pattern, is different in meta-analytical models performed through `mixmeta` if compared to other regression functions such as `lm` or `glm`, in particular for the multivariate case. Specifically, while a unit is removed if at least an observation for one predictor is missing, partially missing outcomes do not prevent the unit to contribute to estimation (see `mixmeta`). Specific methods `na.omit` and `na.exclude` for class `"data.frame.mixmeta"` allow this different definition.

Value

These functions returns the model frame `object` with rows corresponding to units with invalid missing pattern being removed. They also add the related `na.action` attribute as explained above.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See na.action, naresid and napredict. See model.frame. See mixmeta-package for an overview of the package and modelling framework.

Examples

```
# INPUT MISSING VALUES IN PREDICTOR AND ONE RESPONSE
data <- berkey98
data[2,1] <- data[4,3] <- NA
data

# RUN THE MODEL
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=data[5:7], data=data, method="ml")

# SUMMARIZE: NOTE THE NUMBER OF STUDIES AND OBSERVATIONS
summary(model)
df.residual(model)

# EXTRACT THE MODEL FRAME WITH na.pass
model.frame(model, na.action="na.pass")
# EXTRACT THE MODEL FRAME WITH na.omit (DEFAULT)
model.frame(model, na.action="na.omit")

# COMPARE WITH DEFAULT METHOD FOR na.omit
frame <- model.frame(model, na.action="na.pass")
na.omit(frame)
class(frame)
class(frame) <- "data.frame"
na.omit(frame)

# WITH na.exclude
residuals(model)
residuals(update(model, na.action="na.exclude"))
```

---

p53 *Mutant p53 Gene and Squamous Cell Carcinoma*

Description

The dataset includes studies providing evidence about whether the presence of mutant p53 tumour suppressor gene is a prognostic factor for patients presenting with squamous cell carcinoma arising from the oropharynx cavity. Unadjusted estimates of log hazard ratios of mutant p53 to normal p53 for disease-free and overall survival, together with the associated variances, are collected from 6 observational studies.

Usage

```
p53
```

Format

A data frame with 6 observations on the following 5 variables:

- `study`: study ID.
- `y1, V1`: estimate and associated variance of the log hazard ratio for disease-free survival.
- `y2, V2`: estimate and associated variance of the log hazard ratio for overall survival.

Details

Only 3 studies provide estimates for disease-free survival. The within-study correlations are not reported in the original studies but are expected to be highly positively correlated. The original data are described in Tandon and colleagues (2010) and used as an example by Jackson and colleagues (2011).

Note

The data provide an example of application of multivariate meta-analysis when the within-study correlations are not known. These correlations can be inputted directly in the `mixmeta` function through the `control` argument. See `mixmeta.control` for details.

Source

Jackson D, Riley R, White IR (2011). Multivariate meta-analysis: Potential and promise. *Statistics in Medicine*. **30**(20);2481–2498.

Tandon S, Tudur-Smith C, Riley RD, et al. (2010). A systematic review of p53 as a prognostic factor of survival in squamous cell carcinoma of the four main anatomical subsites of the head and neck. *Cancer Epidemiology, Biomarkers and Prevention*. **19**(2):574–587.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS OF EXAMPLE 3 IN JACKSON ET AL. (2011)

# INSPECT THE DATA
p53

# REML MODEL WITH INPUTTED CORRELATION EQUAL TO 0.95
model <- mixmeta(cbind(y1,y2), cbind(V1,V2), data=p53, control=list(Scor=0.95))
print(summary(model), digits=2)
```

predict.mixmeta *Predicted Values from mixmeta Models*

Description

This method function computes predictions from fitted univariate or multivariate meta-analytical models represented in objects of class "mixmeta", optionally for a new set of predictor values in meta-regression models. Predictions are optionally accompanied by standard errors, confidence intervals or the entire (co)variance matrix of the predicted outcomes.

Usage

```
## S3 method for class 'mixmeta'
predict(object, newdata, se=FALSE, ci=FALSE, vcov=FALSE, ci.level=0.95,
  format, aggregate="stat", na.action=na.pass, ...)
```

Arguments

object	an object of class "mixmeta".
newdata	An optional data frame in which to look for variables values with which to predict from meta-regression models.
se	logical switch indicating if standard errors must be included.
ci	logical switch indicating if confidence intervals must be included.
vcov	logical switch indicating if the (co)variance matrix must be included.
ci.level	a numerical value between 0 and 1, specifying the confidence level for the computation of confidence intervals.
format	the format for the returned results. See Value.
aggregate	when format="matrix" and se or ci are required, the results may be aggregated by statistic or by outcome. See Value
na.action	a function which indicates what should happen when the data contain NAs. The default to the value saved in object. See Note.
...	further arguments passed to or from other methods.

Details

The method function predict produces predicted values from mixmeta objects, obtained by evaluating the original call to mixmeta in the frame newdata. For both fixed and random-effects models, estimated predictions are only based on the fixed part of the model, ignoring study-specific deviations, differently from blup.

If newdata is omitted, the predictions are based on the data used for the fit. In that case how to handle predictions for units removed from estimation due to invalid missing pattern is determined by the na.action argument used in mixmeta to produce object. If na.action=na.omit, units excluded from estimation will not appear, whereas if na.action=na.exclude they will appear, with values set to NA for all the outcomes. This step is performed by napredict. See Notes.

Value

The results may be aggregated in matrices (the default), or returned as lists, depending on the argument `format`. For multivariate models, the aggregation is ruled by the argument `aggregate`, and the results may be grouped by statistic or by outcome. If `vcov=TRUE`, lists are always returned.

Note

The definition of missing in model frames used for estimation in `mixmeta` is different than that commonly adopted in other regression models such as `lm` or `glm`. See info on `missing values` in `mixmeta`.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See `blup` for best linear unbiased predictions. See the default method `predict`. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98)

# PREDICTED FROM YEAR 1985 TO 1987, WITH LABELS
newdata <- data.frame(pubyear=1985:1987, row.names=1985:1987)

# AVERAGED OUTOCOMES AND SE
predict(model, newdata, se=TRUE)

# SAME AS ABOVE, AGGREGATED BY OUTCOME
predict(model, newdata, se=TRUE, aggregate="outcome")

# WITH VCOV, FORCED TO A LIST
predict(model, newdata, se=TRUE, vcov=TRUE, aggregate="outcome")
```

qtest *Cochran Q Test of Heterogeneity*

Description

This is a generic function to perform a Cochran Q test of (residual) heterogeneity. The function invokes particular `methods` which depend on the `class` of the first argument. Currently, specific methods exist for several meta-analytical models in various packages: `qtest.mixmeta`, `qtest.mvmeta`, and `qtest.dosresmeta`.

Usage

```
qtest(object, ...)
```

Arguments

`object`	an object for which the test is desired
`...`	further arguments passed to specific methods.

Details

The test assesses the null hypothesis that the variability in the distribution of the outcomes is explained only in terms of within-unit estimation errors. This corresponds to a test on the hypotesis that there is no variation attributable to random-effects terms.

Value

Returned values depend on the specific class. Usually, the results of the test.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

References

Cochran WG (1950). The comparison of percentages in matched samples". *Biometrika*. **37**(3/4):256–266.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

Specific methods for various classes: `qtest.mixmeta`, `qtest.mvmeta`, and `qtest.dosresmeta`.

`qtest.mixmeta` *Cochran Q Test of Heterogeneity for mixmeta Models*

Description

This method function performs a Cochran Q test of (residual) heterogeneity on fitted meta-analytical models represented in objects of class `"mixmeta"`.

Usage

```
## S3 method for class 'mixmeta'
qtest(object, ...)

## S3 method for class 'qtest.mixmeta'
print(x, digits=3, ...)
```

Arguments

`object`, `x`	objects of classes `"mixmeta"` and `"qtest.mixmeta"`, respectively.
`digits`	an integer specifying the number of digits to which printed results must be rounded.
`...`	further arguments passed to or from other methods.

Details

The test assesses the null hypothesis that the variability in the distribution of the outcomes is explained only in terms of estimation error in each unit, measured by the within-unit (co)variance matrices stored in the component `S` of `mixmeta` objects. This is equal to test the hypothesis that the random-effects (co)variance matrix (or all matrices in multilevel models) is a zero matrix, and there is no random deviation in unit-specific estimates. For multivariate models, tests for single outcome parameters, comparable to estimates from multiple univariate meta-analysis, are also reported. This test reduces to the standard Q test in univariate single-level models.

The function compute the statistics by actually fitting the related fixed-effects model, re-evaluating the call of the model with `method` changed to `"fixed"`.

Value

A list object of class `"qtest.mixmeta"` with the following components:

`Q`	the vector of test statistics for overall and outcome-specific tests, distributed under the null hypothesis as a Chi-square with degrees of freedom `df`.
`df`	the vector of degrees of freedom of the null distribution for overall and outcome-specific tests. For the overall test, equal to the number of observations used for estimation minus the number of coefficients in the fixed part of the model. For outcome-specific test, equal to number of observed values minus the number of coefficients.

`pvalue`	the vector of p-values for overall and outcome-specific tests.
`residual`	logical switch indicating if a meta-regression model is assessed, meaning that the tested heterogeneity is residual.
`k`	dimensionality of the overall test, that is the number of outcome parameters in the model.

As usual, the `print` method function for class `"qtest.mixmeta"` does not return any value.

Note

In multivariate models, tests on single outcome parameters are performed by extracting the related estimates and variances, but they do not account for the correlation between them, which nevertheless has been considered in estimation. These tests are not therefore comparable with those performed by running a univariate model on each outcome parameter.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Cochran WG (1950). The comparison of percentages in matched samples". *Biometrika*. **37**(3/4):256–266.

See Also

See `qtest` for the generic method function. See `mixmeta-package` for an overview of the package and modelling framework.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL) ~ 1, S=berkey98[5:7], data=berkey98)

# MULTIVARIATE COCHRAN Q TEST FOR HETEROGENEITY
test <- qtest(model)
print(test, digits=2)
unclass(test)
```

school *Studies on Modified School Calendar and Student Achievement*

Description

The dataset contains the results of 56 studies that evaluate the effect of a modified school calendar on student achievement. The studies assessed students from grade 1 to 9 and reported standardized reading achievement differences between schools that follow a year-round versus the traditional nine-month calendar. The studies were performed in separate school districts, with at least three studies in each district.

Usage

```
school
```

Format

A data frame with 56 observations on the following 5 variables:

- `district, study`: numbers identifying the school district and study, respectively.
- `effect`: estimated standardized effect, reported as difference in reading achievement expressed in standard deviation units.
- `var`: within-study variance of the estimated effects.
- `year`: year when the study was performed.

Note

The data provide an example of application of multilevel meta-analysis with multiple nested random-effects levels, where effect sizes are correlated between studies within school district. This more complex correlation structure is modelled by two levels of random effects. Results can be compared with the so-called three-level model in Kostantopoulos (2011), that is defined as a two-level meta-analysis here.

Source

Kostantopoulos S (2011). Fixed effects and variance components estimation in three-level meta-analysis. *Research Synthesis Methods*. **2**(1):61–76.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN KOSTANTOPOULOS (2011), TABLES 4 AND 5

# STANDARD META-ANALYSIS (NB: random NOT STRICTLY NEEDED HERE)
mod1 <- mixmeta(effect, var, random= ~ 1|study, data=school, method="ml")
print(summary(mod1), digits=3, report="var")
```

```
# STANDARD META-REGRESSION
yearcen <- school$year - mean(school$year)
mod2 <- mixmeta(effect ~ yearcen, var, random= ~ 1|study, data=school,
  method="ml")
print(summary(mod2), digits=3, report="var")

# TWO-LEVEL META-ANALYSIS
mod3 <- mixmeta(effect, var, random= ~ 1|district/study, data=school,
  method="ml")
print(summary(mod3), digits=3, report="var")

# TWO-LEVEL META-REGRESSION
yearcen2 <- with(school, year - mean(tapply(year, district, mean)))
mod4 <- mixmeta(effect ~ yearcen2, var, random= ~ 1|district/study, data=school,
  method="ml")
print(summary(mod4), digits=3, report="var")

### SEE help(thrombolytic) FOR A COMPLEMENTARY EXAMPLE
```

`smoking` *Meta-Analysis of Interventions to Promote Smoking Cessation*

Description

The dataset contains the results of 24 trials comparing four alternative interventions to promote smoking cessation. The trials have different designs, comparing two or three different interventions. The data consist of the number of successes out of the total participants, and the estimated log-odds ratio for arms B (self-help), C (individual counselling), and D (group counselling) relative to arm A (no contact), as well as the (co)variance matrix of these three estimates.

Usage

```
smoking
```

Format

A data frame with 24 observations on the following 19 variables:

- `study`: study ID.
- `design`: design of the trial, reporting the interventions being compared.
- `dA`, `dB`, `dC`, `dD`: number of successes for each intervention.
- `nA`, `nB`, `nC`, `nD`: number of participants for each intervention.
- `yB`, `yC`, `yD`: estimated log-odds ratios for interventions B, C and D versus intervention A.
- `SBB`, `SBC`, `SBD`, `SCC`, `SCD`, `SDD`: variances and co-variances of the estimated log-odds ratios for interventions B, C and D versus intervention A. The order corresponds to the lower triangular elements of the (co)variance matrix taken by column.

Details

Intervention A is chosen as the reference category. Trials without an arm A (trials 2 and 21-24) are augmented with an arm A with 0.01 individuals and 0.001 successes. Trials containing zero cells (trials 9 and 20) have 1 individual with 0.5 successes added to each intervention. Details on the data augmentation and estimation of (co)variances of the log-odds ratios are provided by White (2011).

Note

The data provide an example of application of network meta-analysis, also referred to as indirect mixed-treatment comparison. Additional information using examples based on these data are provided by Lu and Ades (2006), White (2011) and Higgins and colleagues (2012).

Source

Lu G and Ades AE (2006). Assessing evidence inconsistency in mixed treatment comparisons. *Journal of the American Statistical Association*. **101**:447–459.

Higgins JPT, et al. (2012). Consistency and inconsistency in network meta-analysis: concepts and models for multi-arm studies. *Research Synthesis Methods*. **3**(2):98–110.

White IR (2011). Multivariate random-effects meta-regression. *The Stata Journal*. **11**:255–270.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN WHITE (2011)

# INSPECT THE DATA
head(smoking)
names(smoking)

# CONSISTENCY MODEL, UNSTRUCTURED BETWEEN-STUDY (CO)VARIANCE
y <- as.matrix(smoking[11:13])
S <- as.matrix(smoking[14:19])
mod1 <- mixmeta(y, S)
summary(mod1)

# CONSISTENCY MODEL, STRUCTURED BETWEEN-STUDY (CO)VARIANCE (PROPORTIONAL)
mod2 <- mixmeta(y, S, bscov="prop", control=list(Psifix=diag(3)+1))
summary(mod2)

# TRANSFORM IN LONG FORMAT, WITH S AS LIST (EXCLUDING MISSING)
long <- na.omit(reshape(smoking[,c(1,2,11:13)], varying=list(3:5), idvar="study",
  v.names="y", timevar="outcome", times=colnames(y), direction="long"))
Slist <- lapply(lapply(seq(nrow(S)), function(i) xpndMat(S[i,])), function(x)
  x[!is.na(diag(x)), !is.na(diag(x)), drop=FALSE])

# THE MODELS ABOVE CAN BE REPLICATED IN THE LONG FORMAT
mod2b <- mixmeta(y ~ 0 + factor(outcome), random= ~ 0 + factor(outcome)|study,
  data=long, bscov="prop", control=list(addS=Slist, Psifix=diag(3)+1))
summary(mod2b)
```

```
# DEFINE AND ADD INDICATORS FOR OUTCOME AND DESIGN
dummy <- cbind(model.matrix(~0+outcome, long), model.matrix(~0+design, long))
colnames(dummy) <- c(levels(factor(long$outcome)), levels(long$design))
long <- cbind(long, data.frame(dummy))

# INCONSISTENCY MODEL (SPECIAL PARAMETERIZATION OF OUTCOME-BY-DESIGN INTERACTION)
formula <- y ~ 0 + yB + yC + yC:acd + yC:bc + yC:bcd + yD + yD:acd + yD:bcd +
  yD:bd + yD:cd
mod3 <- update(mod2b, formula=formula)
summary(mod3)
```

`summary.mixmeta` *Summarizing mixmeta Models*

Description

Print and summary method functions for fitted meta-analytical models represented in objects of class `"mixmeta"`.

Usage

```
## S3 method for class 'mixmeta'
summary(object, ci.level=0.95, ...)

## S3 method for class 'summary.mixmeta'
print(x, digits=4, report=c("sd","var"), ...)

## S3 method for class 'mixmeta'
print(x, digits=4, ...)
```

Arguments

`object`	an object of class `"mixmeta"` produced by a call to `mixmeta`.
`x`	an object of class `"mixmeta"` or `"summary.mixmeta"`, produced by calls to `mixmeta` or `summary.mixmeta`, respectively.
`ci.level`	a numerical value between 0 and 1, specifying the confidence level for the computation of confidence intervals.
`digits`	an integer specifying the number of digits to which printed results must be rounded.
`report`	if standard deviations (`sd`) or variances (`var`) must be reported for summarizing the random-effects (co)variance structure.
`...`	further arguments passed to or from other methods.

Details

The print method function for class "mixmeta" only returns basic information on the fitted model, namely the call, estimated fixed-effects coefficients, dimensions and fit statistics (log-likelihood, AIC, BIC).

The summary method function computes additional statistics and tests, and produces a list object of class "summary.mixmeta". The print method function for this class shows additional information, such as tables reporting the estimates for the fixed and random-effects parts of the model, Cochran Q test for heterogeneity and I-squared.

Value

The summary method function for mixmeta objects produces a list of class "summary.mixmeta". The components of the lists are some of those stored in the related mixmeta object, plus the following:

coefficients	a matrix reporting point estimates, standard errors, z statistics and related p-values of the test, and confidence intervals for the kp fixed-effects coefficients. Note this is different than the component with the same name stored in mixmeta objects, simply reporting the point estimates (see mixmetaObject).
AIC	the value of the Akaike information criterion for the fitted mixmeta model, obtained through a call to AIC.
BIC	the value of the Bayesian information criterion for the fitted mixmeta model, obtained through a call to BIC.
corFixed	the $kp \times kp$ correlation matrix of the fixed-effects coefficients, obtained from the (co)variance matrix vcov (see mixmetaObject and vcov).
corRandom	the $kq \times kq$ correlation matrix of the random effects, obtained from the random-effects (co)variance matrix Psi, or a list of multiple matrices for multilevel models. See mixmetaObject.
qstat	results from the Cochran Q test for heterogeneity, namely a list corresponding to a qtest.mixmeta object without its class, obtained through qtest.
i2stat	I-squared statistic for the meta-analytical model.
ci.level	the confidence level used for defining the confidence intervals for the estimates of the fixed-effects coefficients.

As usual, the print method functions for classes "mixmeta" and "summary.mixmeta" do not return any value.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>> and Francesco Sera <<francesco.sera@lshtm.ac.uk>>

References

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

See Also

See mixmeta and mixmetaObject.

Examples

```
# RUN THE MODEL
model <- mixmeta(cbind(PD,AL) ~ pubyear, S=berkey98[5:7], data=berkey98)

# SIMPLE PRINT
model

# DEFINE DIGITS
print(model, digit=2)

# SUMMARY WITH 80TH CONFIDENCE INTERVALS
summary(model, ci.level=0.80)

# REPORT RANDOM EFFECTS IN TERMS OF VARIANCES (USE print)
print(summary(model), report="var")
```

---

terms.mixmeta *Extract Model Terms from mixmeta Objects*

Description

These method function returns the terms object that defines meta-analytical models represented in objects of class "mixmeta".

Usage

```
## S3 method for class 'mixmeta'
terms(x, type="fixed", ...)
```

Arguments

x	an object of class "mixmeta".
type	the type of terms. Either "fixed" or "full". See Details.
...	further arguments passed to or from other methods.

Details

The terms object is produced by mixmeta when fitting the meta-analytical model, and stored as an attribute of the model.frame. Note that this object consists of terms for both the fixed and random-effects parts, the latter including also the grouping factors.

By using the default type="fixed", this method function removes the random-effects terms. This can then be used, for instance, for creating the model.matrix for the fixed effects. Otherwise with type="full", the full set of terms is returned.

Value

An object of class `c("terms","formula")` which contains the terms representation of a symbolic meta-analytical model. See `terms.object` for its structure.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See the methods `model.frame` and `model.matrix`.

Examples

```
# RUN A MODEL
model <- mixmeta(effect, var, random= ~ 1|district/study, data=school)

# TERMS (FIXED AND FULL)
terms(model)
terms(model, "full")
attr(model.frame(model), "terms")
```

---

thrombolytic *Randomized Trials of Thrombolytic Therapy*

Description

The dataset contains the data on 20 randomized trials of thrombolytic therapy, which evaluated effect on short-term mortality after a myocardial infarction (up to 35 days) in 50,246 patients in relation to treatment delay. The hypothesis is that the thrombolytic therapy reduces the mortality risk following the myocardial infarction, and that the benefit is particularly substantial for very early treatment. Some of the trials report separate results according to treatment delay, generating 38 observations from full trials or subgroups of trials. Effect sizes were reported as absolute risk reduction computed as the difference between treated and control groups in each trial or subgroup.

Usage

```
thrombolytic
```

Format

A data frame with 38 observations on the following 10 variables:

- `trial`: label identifying the trial.
- `time2treat`: treatment delay after the onset of the symptoms of a myocardial infaction, reported in hours.
- `dtreat`, `ntreat`: number of deaths and total patients in the treated group, respectively.

- `dcontr`, `ncontr`: number of deaths and total patients in the control group, respectively.
- `risktreat`, `riskcontr`: risk of death in the treatment and control groups, respectively.
- `absrisk`: absolute risk difference of death between the treatment and control groups. See Details.
- `var`: variance of the absolute risk difference. See Details.

Details

The absolute risk is simply the difference in risk, which is computed empirically as ratio of the number of deaths and the number of total patients in treated and control groups ($p_1 = d_1/N_1$ and $p_0 = d_0/N_0$, respectively). The variance of the absolute risk difference is computed as $p_0(1 - p_0)/N_0 + p_1(1 - p_1)/N_1$. See Thompson and colleagues (2001) for details.

Note

The data provide an example of application of multilevel meta-analysis with repeated observations in an inner level within an outer level, corresponding here to treatment subgroups within each trial. This more complex correlation structure is modelled by two levels of random effects, including meta-predictors that can explain part of the heterogeneity at each level. Results can be compared with those reported by Thompson and colleagues (2001).

Source

Thompson SG, Turner RM, Warn DE (2001). Multilevel models for meta-analysis, and their application to absolute risk differences. *Statistical Methods in Medical Research*. **10**(6):375–392.

Sera F, Armstrong B, Blangiardo M, Gasparrini A (2019). An extended mixed-effects framework for meta-analysis.*Statistics in Medicine*. 2019;38(29):5429-5444. [Freely available **here**].

Examples

```
### REPRODUCE THE RESULTS IN THOMPSON ET AL (2001), TABLES 2, 3, AND 4

# STANDARD FIXED-EFFECTS META-ANALYSIS
mod1 <- mixmeta(absrisk, var, data=thrombolytic, method="fixed")
print(summary(mod1), digits=5)

# STANDARD RANDOM-EFFECTS META-ANALYSIS
subtrial <- seq(nrow(thrombolytic))
mod2 <- mixmeta(absrisk, var, random= ~ 1|subtrial, data=thrombolytic)
print(summary(mod2), digits=5)

# TWO-LEVEL RANDOM-EFFECTS META-ANALYSIS
mod3 <- mixmeta(absrisk, var, random= ~ 1|trial/subtrial, data=thrombolytic)
print(summary(mod3), digits=5)

# TWO-LEVEL RANDOM-EFFECTS META-REGRESSION
mod4 <- mixmeta(absrisk~time2treat, var, random= ~ 1|trial/subtrial,
  data=thrombolytic)
print(summary(mod4), digits=5)
```

```
# TWO-LEVEL RANDOM-EFFECTS META-REGRESSION WITH NON-LINEAR TERM
mod5 <- mixmeta(absrisk ~ time2treat + I(1/time2treat), var,
  random= ~ 1|trial/subtrial, data=thrombolytic)
print(summary(mod5), digits=5)

### SEE help(school) FOR A COMPLEMENTARY EXAMPLE
```

---

vechMat *Vectorization and Expansion of Symmetric Matrices*

Description

The function vechMat transforms a symmetric matrix in a vector containing its lower triangular elements, taken by column. The function xpndMat reverses this transformation.

Usage

```
vechMat(mat, diag=TRUE)

xpndMat(vech)
```

Arguments

mat	a square matrix.
vech	a vector.
diag	a logical switch indicating if the diagonal entries must be included.

Value

A vector for vechMat, a symmetric matrix for xnpdMat.

Author(s)

Antonio Gasparrini <<antonio.gasparrini@lshtm.ac.uk>>

See Also

See functions vech and xpnd in package **MCMCpack**.

Examples

```
# GENERATE A POSITIVE-DEFINITE MATRIX, VECTORIZE IT AND THEN RE-EXPAND
(M <- crossprod(matrix(rnorm(9),3)))
(v <- vechMat(M))
xpndMat(v)

# EXTRACT VECTORIZED S, EXPAND TO A LIST, AND RE-VECTORIZE
(S <- as.matrix(berkey98[5:7]))
(Slist <- lapply(seq(nrow(S)), function(i) xpndMat(S[i,])))
t(sapply(Slist,vechMat))
```

Index