Package 'dmm'

June 3, 2024

Type Package

Title Dyadic Mixed Model for Pedigree Data

Version 2.1-10

Date 2024-05-31

Author Neville Jackson

Maintainer Neville Jackson <nanddjackson@bigpond.com>

Description Dyadic mixed model analysis with multi-trait responses and pedigree-based partitioning of individual variation into a range of environmental and genetic variance components for individual and maternal effects. Method documented in dmmOverview.pdf; dmm is an implementation of dispersion mean model described by Searle et al. (1992) ``Variance Components'', Wiley, NY.

Depends MASS, Matrix, robustbase, pls, nadiv

Imports stats, graphics, grDevices

License GPL-2 | GPL (>= 2) | GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-06-03 06:30:03 UTC

R topics documented:

`dmm-package` *Dyadic mixed model analysis for pedigree data*

Description

Dyadic mixed model analysis with multi-trait responses and pedigree-based partitioning of an individual random effect into a range of genetic and environmental (co)variance components for individual (ie direct) and maternal contributions to phenotype.

Details

Package:	dmm
Type:	Package
Version:	2.1-8
Date:	2023-07-13
License:	GPL-2

This package provides tools for setting up and solving dyadic model equations leading to estimates of variance components and their standard errors, for transforming variance components to genetic parameters and their standard errors, and for computing genetic response to selection.

You may wish to use this package if you are looking for any of the following features in a quantitative genetic analysis:

- suited to small multi-trait datasets with pedigree information
- individual, maternal, and cohort environmental component estimates and standard errors
- individual and maternal additive, dominance, epistatic, and sex-linked genetic component estimates and standard errors
- cross-effect and cross-trait covariance components
- multicollinearities among the components
- genetic parameters (ie proportion of variance and correlation) and standard errors for all fitted components
- genetic response to phenotypic selection for individual additive and maternal additive cases with autosomal and sexlinked components
- data preparation tools
- S3 methods to organize output
- test example datasets
- alternative approach to iterative ML and REML estimation procedures
- component estimates equivalent to MINQUE (after fixed effects by OLS) and bias-corrected-ML (after fixed effects by GLS)
- multi-trait or traitspairwise or traitsblockwise analyses
- class-specific genetic parameters
- maternal or paternal founderline components

The main functions in dmm are:

dmm() Sets up and solves dyadic model equations for a dataset which is supplied as a dataframe containing both the pedigree information and the observations

mdf() Checks the dataframe for compliance with dmm requirements, converts multi-trait data to a matrix within the dataframe, and optionally appends relationship matrices to the dataframe.

summary() S3 method, reports estimated (co)variance components and standard errors

csummary() S3 method, reports reports (co)varianve components with standard erors, sorted into class-specific groups, so that thaey sum to phenotypic (co)variance within each group

gsummary() S3 method, reports genetic parameters and standard errors

gresponse() S3 method, reports genetic response to selection

print() S3 method, briefly reports output object from dmm()

plot() S3 method, plots residuals for dyadic model fit

There are also some example datasets, some with 'known' answers:

dt8bal.df A small balanced dataset showing agreement with aov in balanced case

harv103.df A real dataset from Harvey(1960) with extensive fixed effects

merino.df A large real multi-trait dataset from a Merino sheep breeding experiment

quercus.df A 2-trait dataset supplied with the QUERCUS program

sheep.df A small 3-trait dataset used for demonstration

tstmo1.df A univariate dataset supplied with the DFREML program

warcolak We also use the warcolak dataset from package `nadiv`

To use dmm one first must put the desired dataset into an R workspace as a dataframe object. The minimum requirement is for a dataframe with columns labelled :

Id Identifier for each individual

SId Identifier for the sire of each individual

DId Identifier for the dam of each individual

Sex Sex code for each individual

Fixed factors Codes for levels of each fixed factor

Observations Numeric values for each observation or trait

There are other requirements, and these are documented under the `mdf()` help page, which also documents how to use `mdf()` to convert the user's dataframe to an acceptable form, which can be either another dataframe or an object of class `mdf`.

Given an acceptable data object, one simply calls function `dmm()` with appropriate arguments, the first of which is the data object's name. There are `formula` arguments to specify fixed effects and cohorts, and the components to be partitioned are specified in a simple vector of names. Arguments are documented under the `dmm()` help page. An object of class `dmm` is returned and should be saved in the R workspace.

Given a `dmm` object, there are S3 methods to display the results as follows:

print() Reports fixed effect coefficient and (co)variance component estimates

summary() Reports fixed effect coefficient and (co)variance component estimates with standard errors and confidence limits

gprint() Reports genetic parameters (proportion of variance and correlation) for each component partitioned

gsummary() Reports genetic parameters with standard errors and confidence limits

gresponse() reports genetic response to selection, for estimated parameters

These functions are documented on their help pages. Other results (eg plots) may be obtained by accessing the `dmm` object's attributes directly. See `dmm.object` help page.

Author(s)

Neville Jackson

Maintainer: Neville Jackson <nanddjackson@bigpond.com>

References

dmmOverview.pdf

See Also

In the dmm package

dmm() for dmm function arguments and return value

summary() for fixed coefficients and (co)variance components

gsummary() for genetic parameters

gresponse() for predicted selection response

make.ctable() for comprehensive list of variance components

mdf() for data preparation

print() for brief print of dmm() output

plot() for residual plots for dyadic model

Other R packages

- pedigreemm
- nadiv
- varComp
- minque
- gremlin

Examples

```
library(dmm)
# simple univariate case, direct from the dataframe
data(dt8bal.df)
dt8.fit <- dmm(dt8bal.df, CWW ~ 1 + Sex,
   components=c("VarE(I)","VarG(Ia)"))
summary(dt8.fit)  # fixed effects and variance components
gsummary(dt8.fit) # heritability with se's
rm(dt8.fit)
rm(dt8bal.df)

# Note: 'dt8bal.df' is a small demo dataset. Results are
#        illustrative but not meaningful.
# for more examples see 'dmm' help page and references
# for a tutorial and fully documented examples see  'dmmOverview.pdf'
```

chartodec *Convert a vector from character to decimal numbers*

Description

A vector of numbers encoded as character strings without a decimal point is converted to numbers with a decimal point in a given position

Usage

```
chartodec(cvec, ndec)
```

Arguments

cvec	A vector of type character.
ndec	An integer giving the number of digits to follow the implied decimal point

Details

It is a common problem when reading a table of fixed width formatted data into a dataframe using function `read.fwf()` for the dataframe columns to end up of type character with no decimal point. If these columns are actually meant to be decimal numbers with an implied decimal point in a fixed position, they can be converted to numeric with this function.

Value

A vector of numerical values with a decimal point inserted as per argument `ndec`

Author(s)

Neville Jackson

See Also

Function `read.fwf()`

Examples

```
library(dmm)
tmp <- c("1","2","3",NA)
ntmp <- chartodec(tmp,1)
str(ntmp)
rm(tmp)
rm(ntmp)
```

condense.dmmarray *Condense an object of type* dmmarray *to an object of type* dmm

Description

An object of type dmmarray is an array of objects of type dmm, with each array element representing the reult of a dmm() analysis for one pair of traits. The function condense.dmmarray will recombine these results into a single object of class dmm with the variance component and genetic parameter estimates matrices being for all traits, and other elements of the dmm object being appropriately pooled.

Usage

```
condense.dmmarray(da)
```

Arguments

da	An object of class dmmarray

Details

In bringing together the results of several traitspairwise analyses into a single matrix of (for example) individual additive genetic variance/covariance components, one is putting together into one matrix elements estimated with different precisions due to differnt replication for each pair of traits. The resulting matrix may not be positive definite even if all the contributing traitspairwise 2 x 2 matrices are forced positive definite.

If the argument da contains results from a dmm run with gls=T then the GLS results for each traitpair will also be condensed. In this case the GLS results must be present for every traitpair. It can be quite difficult to get gls=T runs to converge successfully for every trait pair.

Value

An object of class dmm, containing the recombined results for all traits.

Note

An object of class dmmarray can be manipulated as is without using the condense.dmmarray() function. For example one element of the array can be printed with

print(objectname[[i,j]])

where i and j are subscripts indicating the row and column position of the element to be printed. The summary() and gsummary() functions are also available for use in this manner. It is necessary that the library(dmm) statement be made, otherwise one will get the standard print() and summary() functions instead of those appropriate for an object of class dmm. The double square brackets are necessary because each element of the array is a list object, and you want its contents, not its name attribute.

Author(s)

Neville Jackson

See Also

Functions dmm(), condense.dmmblockarray()

Examples

```
library(dmm)
# prepare the dataset sheep.df
data(sheep.df)
# add a matrix 'Ymat' to the dataframe,
#    which is required for traitspairwise
# keep=TRUE is required
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),keep=TRUE)
# make sheep.fit as a class dmmarray object
sheep.fit <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"),traitspairwise=TRUE)
#  look at one element of the dmmarray
summary(sheep.fit[["Cww","Diam"]])
# condense the dmmarray to a class dmm object
sheep.condense <- condense.dmmarray(sheep.fit)
# compute a response to selection
sheep.resp <- gresponse(sheep.condense,
       psd=list(dp=c(1,1,1)),effects=c("G(Ia)"))
# look at response object
summary(sheep.resp)
#cleanup
rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fit)
rm(sheep.condense)
rm(sheep.resp)
```

---

condense.dmmblockarray

Condense an object of type dmmblockarray *to an object of type* dmm

---

Description

An object of class dmmblockarray is a list containing two items called array and blocks. Item array is an array of objects of type dmm, with each array element representing the reult of a dmm() analysis for one pair of blocks of traits. The function condense.dmmarray will recombine these results into a single object of class dmm with the variance component and genetic parameter estimates matrices being for all traits, and other elements of the dmm object being appropriately pooled. Item blocks is a list, each element of which is a list of the traits present in each block.

Usage

```
condense.dmmblockarray(da)
```

Arguments

da	An object of class dmmblockarray

Details

In bringing together the results of several traitsblockwise analyses into a single matrix of (for example) individual additive genetic variance/covariance components, one is putting together into one matrix elements estimated with different precisions due to differnt replication for each block of traits. The resulting matrix may not be positive definite even if all the contributing traitsblockwise ni x nj matrices are forced positive definite.

If the argument da contains results from a dmm run with gls=T then the GLS results for each traitblockpair will also be condensed. In this case the GLS results must be present for every traitblockpair. It can be quite difficult to get gls=T runs to converge successfully for every pair of blocks.

Value

An object of class dmm, containing the recombined results for all traits.

Note

An object of class dmmblockarray can be manipulated as is without using the

condense.dmmblockarray function. For example one element of the array can be printed with print(objectname$array[[i,j]])

where i and j are subscripts indicating the row and column position of the element to be printed. The summary() and gsummary() functions are also available. It is necessary that the

library(dmm) statement be made, otherwise one will get the standard print() and summary() functions instead of those appropriate for an object of class dmm. The double square brackets are necessary because each element of the array is a list object, and you want its contents, not its name attribute.

Author(s)

Neville Jackson

See Also

Functions dmm(), condense.dmmarray()

Examples

```
library(dmm)
# prepare the dataset sheep.df
data(sheep.df)
# add a matrix 'Ymat' to the dataframe,
#         which is required for traitsblockwise
```

```
# keep=TRUE also required
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),keep=TRUE)
# make sheep.fit as a class dmmarray object
sheep.fit <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"),traitsblockwise=TRUE,
    Block1=c("Cww","Diam"),Block2="Bwt")
#  look at one element of the dmmblockarray
summary(sheep.fit$array[["Block1","Block2"]])
# condense the dmmblockarray to a class dmm object
sheep.condense <- condense.dmmblockarray(sheep.fit)
# compute a response to selection
sheep.resp <- gresponse(sheep.condense,
       psd=list(dp=c(1,1,1)),effects=c("G(Ia)"))
# look at response object
summary(sheep.resp)
#cleanup
rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fit)
rm(sheep.condense)
rm(sheep.resp)
```

`csummary.dmm` *Make summary tables of (co)variance component estimates sorted into class-specific classes for a dmm object.*

Description

Extracts the (co)variance component estimates from an object of class `dmm`, for the specified set of traits and set of components. Makes tables of component estimates ordered either by trait or by component. Tables include component estimate, its standard error, and its 95 percent confidence limits. Components are grouped into class-specific classes, if there are any class-specific components fitted.

Usage

```
## S3 method for class 'dmm'
csummary(object, traitset = "all", componentset = "all", bytrait = T,
            gls = F, digits = 3, ...)
```

Arguments

`object`	An object of class `dmm`. (Co)variance component estimates are obtained from this object.
`traitset`	A vector containing the names of the subset of traits for which tables of (co)variance component estimates are to be constructed. Default is "all" which means all traits present in object `object`.

componentset	A vector containing the names of the subset of (co)variance components for which tables are to be constructed. Default is "all" which means all (co)variance components present in object object.
bytrait	Logical flag: should the tables of (co)variance component estimates be constructed with trait varying least rapidly from line to line? If TRUE each subtable contains component estimates for one trait or traitpair and for all components in argument componentset. If FALSE each subtable contains component estimates for one component and for all traits or traitpairs.
gls	Logical flag: should the (co)variance component estimates by GLS-b method be tabled in addition to the (co)variance component estimates by OLS-b method? Default is gls=FALSE. The GLS-b (co)variance component estimates can only be tabled if object object contains the attribute gls, that is if object was constructed by a dmm() call with argument gls=TRUE.
digits	Number of digits for output. This is returned as part of the return value for us e by the S3 print function print.csummary.dmm().
...	Ellipsis argument.

Details

This is a long printout with estimates, standard errors and confidence limits, arranged in tables with one estimate per line. For a short printout see function print.dmm(). In the case of class-specific components, the components are listed in class groups, so that they sum to the class phenotypic variance within each group. In the case where all components are non-specific, there is just one class group.

Value

An object of class csummary.dmm which is a list containing the following items:

ctables	A list of dataframe objects each containing one subtable of estimates of the (co)variance components, along with the appropriate standard errors and confidence limits. Based on OLS-b component estimates.
gctables	A list of dataframe objects each containing one subtable of estimates of the (co)variance components, along with the appropriate standard errors and confidence limits. Based on GLS-b component estimates. Only present if argument gls=TRUE.
traits	A vector of traitnames as specified in argument traitset.
components	A vector of component names as specified in argument componentset.
bytrait	Logical flag: as specified in argument bytrait.
gls	Logical flag: as specified in argument gls.
digits	A numeric value, as specified in argument digits.
call	The function call

Note

There is no provision to constrain the 95 percent confidence limits for component estimates. Hence for small samples, these may vary outside the bounds for the component, that is for components which are variances, they may be negative. Fixed effects are not bounded. Use `csummary()` if you want to see the components summing to phenotypic (co)variance, and sorted into class-specific groups. Use `summary()` if you just want all the components as estimated.

Author(s)

Neville Jackson

See Also

Function `print.csummary.dmm()`.

Examples

```
# get some data
data(sheep.df)
# prepare it - only need "E" and "A" relationship matrices
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# estimate (co)variance components  - VarG(Ia) is Sex-specific
sheep.fitc <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
              components=c("VarE(I)"),
              specific.components=list(Sex=c("VarG(Ia)")))
# look  at components within Sex classes
csummary(sheep.fitc,bytrait=FALSE)
# look just at trait "Cww"
summary(sheep.fitc,traitset="Cww")
# cleanup
rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fitc)
```

---

dmm *Fit a dyadic mixed model to pedigree data*

Description

Dyadic mixed model analysis with multi-trait responses and pedigree-based partitioning of individual variation into a range of environmental and genetic variance components for individual and maternal effects.

Usage

```
dmm(mdf, fixform = Ymat ~ 1, components = c("VarE(I)", "VarG(Ia)"),
    specific.components=NULL, cohortform = NULL, posdef = T, gls = F,
    glsopt = list(maxiter = 200, bdamp = 0.8, stoptol = 0.01),
    dmeopt = "qr", ncomp.pcr = "rank", relmat = "inline", dmekeep = F,
    dmekeepfit = F, traitspairwise=F, traitsblockwise=F,...)

## Default S3 method:
dmm(mdf, fixform = Ymat ~ 1,
    components = c("VarE(I)", "VarG(Ia)"),
    specific.components=NULL, cohortform = NULL, posdef = T, gls = F,
    glsopt = list(maxiter = 200, bdamp = 0.8, stoptol = 0.01),
    dmeopt = "qr", ncomp.pcr = "rank", relmat = "inline", dmekeep = F,
    dmekeepfit = F, traitspairwise=F, traitsblockwise=F,...)
```

Arguments

`mdf` Either a dataframe or an object of class `mdf` which is a list containing a dataframe and one or more relationship matrices, as made by function `mdf()` If missing the variables are searched for in the standard way.

`fixform` A formula specifying the fixed-effect part of the model and the response variate(s). Response should be a matrix for multi-trait models. Default is `Ymat ~ 1` that is a response matrix called `Ymat` and a fitted mean effect.

`components` A simple vector specifying each of the components to be partitioned from the residual variation after fitting fixed effects. Residual is assumed to be the level of variation attributed to individuals. The components given here are assumed to sum to phenotypic variance so that if there are cross-effect covariances (eg "CovG(Ia,Ma)" and "CovG(Ma,Ia)") they need to be both present. The default is `c("VarE(I)", "VarG(Ia)")`, that is a simple partitioning into individual environmental ("VarE(I)") and individual additive genetic ("VarG(Ia)") variation. It is necessary to specify the individual environmental variance, as it is actually fitted as a parameter in the dyadic model, not obtained from residual variation. For a full list of available components see `make.ctable()` function. For a brief introduction to the notation for components see Note section below. For a comprehensive definition of notation see 'dmmOverview.pdf' Section 6.

`specific.components`

A list specifying the name of each specific factor and the variance components which are to be specific to that factor. The list takes the form

`list(Factor1=c(component1,component2,...),`

`Factor2=c(component3,component4,...),...)`.

The default is `NULL`, that is no class specific components are partitioned . Each specific factor must exist as a column in the dataframe. The specific factors do not have to be fitted as fixed effects, but can be. The classes (ie levels) within each factor must be mutually exclusive, that is, each individual can only belong to one level of each factor. For a full coverage of class specific components see the document *dmmClassSpecific.pdf*.

If one wishes all of the specified components to be class specific, one needs to specify `components=NULL` in order to cancel out the default for the `components=` argument.

`cohortform` A formula specifying the effects which define cohort grouping of individuals. For example `cohortform = ~ Year`. A cohort is a grouping of individuals experiencing the same external environmental conditions, eg a group of sheep born and reared together, commonly referred to as a drop. Cohort should not contain DId - the dam's Id. If one needs to consider littermates, function `dmm()` provides a means of combining maternal environmental and cohort variance components to achieve this. The default is NULL - ie no cohort defined.

`posdef` A logical flag: should the matrices of variance components be constrained to be positive definite? If TRUE each matrix of cross-trait (co)variances for each "Varxxx" component defined in `components` will be individually positive definite, and each cross-effect covariance (if "Covxxx" components are defined) will be constrained such that the corresponding correlation is in the bounds -1 to 1. If FALSE all components will be as estimated. The default is TRUE.

`gls` A logical flag: should `dmm()` go on after fitting fixed effects by OLS and estimating components, to re-fit the fixed effects by GLS and re-estimate the components. If TRUE the option `posdef=T` is enforced, as the GLS iteration will fail if matrixes do not remain positive definite. Default is FALSE - ie do the OLS analysis only.

`glsopt` A list object containing variables used to control the GLS iteration :

maxiter Maximum number of iterations. Default 200.

bdamp Factor used to damp the setting of new GLS-b coefficients at each round of iteration. A value of 1.0 means no damping.

stoptol Value below which the sum of absolute deviations of new from old coefficients must fall to achieve convergence.

The GLS iteration normally converges very rapidly. If it does not, consider changing the model, before fiddling with these parameters.

`dmeopt` One of four regression techniques used to solve the dyadic model equations (DME's) to estimate components:

"qr" The default option is to use the QR algorithm directly on the dyadic model equations. This is most efficient, but does not produce a *fit* object for looking at further statistics such as with `anova` or `plot` or `resid`.

"lm" This option calls the `lm()` function to solve the DME's. This is equivalent to QR, but `lm()` produces a *fit* object which can optionally be part of the returned `dmm` object.

"lmrob" This option calls the `lmrob()` function from package `robustbase` to solve the DME's. Robust regression only works for single-trait models. A *fit* object can be returned.

"pcr" This option calls the `mvr()` function from package `pls` with argument `method="svdpc"`. Principal component regression is intended to be used where there are multicollinearities among the components to be estimated. The number of principal components is set to the rank of the DME matrix, but can be overwritten (see `ncomp.pcr` argument). A *fit* object can be returned.

If gls=TRUE the same dmeopt option is also used during the GLS iteration.

ncomp.pcr — Number of principal components to use during a principal components regression (see dmeopt argument). Default is the rank of the DME matrix

relmat — One of two ways of setting up the relationship matrices required to estimate the variance components:

"inline" The additive genetic relationship matrix will be calculated by inline code each time dmm() is run. OK for small datasets. Do not use if non-additive relationship matrices are required. This is the default.

"withdf" The required relationship matrices are assumed to be pre-stored in the object of class mdf defined in the first argument. See function mdf() for setting up relationship matrices in the mdf object. Function mdf() makes extensive use of the package nadiv.

dmekeep — Logical flag: should the dyadic model equations be returned as part of the dmm() return object? Default is FALSE. The DME's may be a large object.

dmekeepfit — Logical flag: should the *fit* object from solving the DME's be returned as part of the dmm() return object. Default is FALSE. The *fit* object may be large.

traitspairwise — Logical flag: should the traits be analysed two at a time in all permutations? Default is FALSE, in which case traits are all analysed simultaneously. This option is useful if traits have different replication. If this option is TRUE dmm() will return an object of class dmmarray, being an array of class dmm objects with the rows and columns named by the trait names. See the Value section for the structure of an object of class dmmarray. For this option the dataframe specified as argument mdf must be made with function mdf() and must contain the matrix of traits 'Ymat' plus the individual traits as columns (keep=T option for mdf()).

traitsblockwise — Logical flag: should the traits be analysed in defined blocks of traits in all permutations of pairs of blocks? Default is FALSE. This option is useful if blocks of traits have different replication. If this option is TRUE, the ellipsis option defining blocks must be present. For this option the dataframe specified as argument mdf must be made with function mdf() and must contain the matrix of traits 'Ymat' plus the individual traits as columns (keep=T option for mdf()).

... — Ellipsis argument: if traitsblockwise is TRUE this argument should contain a number of block definitions of the form

`Block1=c("Trait1","Trait2"),Block2=c("Trait3","Trait4"),...`

, specifying the traits to be present in each block. In this case dmm() will return an object of class dmmblockarray, being an array of class dmm objects with the rows and columns named "Block1" and "Block2", together with lists of the trait names present in each block. See the Value section for the structure of an object of class dmmblockarray.

Details

The minimum requirement to use dmm() directly on a simple dataframe is that it contain columns named "Id", "SId", "DId", and "Sex" plus any fixed effects and traits. The "Id" column must contain identifiers which are unique, numeric, and sequential (ie they must be numbered 1 to n with unit increments, no duplicates and no gaps). Any fixed effects must be factors, and traits must be

numeric. Every "SId" and "DId" code must appear also in the "Id" column even if this results in NA's in every other column. If these requirements are not met, process the dataframe with `mdf()` before using `dmm()` Also if any relationship matrix other than additive is required, pre-processing with `mdf()` is necessary.

Missing values for either traits or fixed effects are simply omitted by `dmm()` before any processing. There is an heirarchy of models fitted by `dmm()`. There is one fixed model and one dyadic model, for all traits, and only individuals for which all traits are present are included in the model fit steps. In contrast, all individuals are included in the pedigree and in setting up relationship matrices. Hence the number of individuals with data, may be less than the number of individuals in the pedigree. If options `traitspairwise` or `traitsblockwise` are used both the fixed model and the dyadic model will be the same for all traitpairs (or traitblocks) but the replication may differ, so missing values for some traits or sets of traits can be handled in this way.

The (co)variance which is partitioned into components is always the residual (co)variance from the fixed effects model. This is assumed to represent the observed variation among individuals. There is, at this stage, no provision for models with more than one error level, so split plot and repeated measures designs are not provided for. There is nothing to stop one formulating the appropriate fixed effects model and doing the `aov()` step, but partitioning of any (co)variance other than residual is not at present provided.

The naming conventions for components may seem a little strange. They are designed to be all ASCII and therefore usable by R as rownames or colnames. The function `make.ctable()` returns a list of all available components (as returnobject$all), as well as a spectrum of sublists which are used internally. The available components are fully documented in the pdf file *dmmOverview.pdf* Section 6. Most of the names are obvious (eg "VarG(Ia)" means variance-genetic-individual-additive). The term *individual* distinguishes individual or direct genetic or environmental effects from maternal genetic or environmental effects.

It is important for the proper estimation of phenotypic (co)variance that any cross-effect covariance components are fitted in symmetric pairs (for example "CovE(I,M)" and "CovE(M,I)"). For one trait these will be identical, so the covariance will simply enter twice in the sum, as required. However cross-trait-cross-effect covariances will not be identical and the sum, which is a phenotypic covariance, requires that the symmetric pair be present.

In addition to the value returned, `dmm()` makes a number of lines of screen output which show each processing step and some minimal model check numbers.

Value

An object of class `dmm` is returned whenever options `traitspairwise` and `traitsblockwise` are both FALSE (ie a normal multi-trait analysis). This object is basically a list of some or all of the following items:

`aov`	An object of class `aov` containing the results of fitting the fixed effects (specified in argument `fixform`) by OLS using a call to the `aov()` function
`mdf`	Not currently used - ignore.
`fixform`	Formula specifying fixed effects fitted.
`b`	Coefficients fitted for fixed effects.
`seb`	Standard errors for fixed effect coefficients.

`vara` Matrix of (co)variances of individuals after adjusting for fixed effects fitted by OLS.

`totn` Total number of individuals in the analysis (ie with data)

`degf` Degrees of freedom remaining after fitting fixed effects.

`dme.wmat` The dyadic model equations matrix. Present only if dmekeep=TRUE. The name 'wmat' refers to the matrix W in equations 12 and 13 of the document dmmOverview.pdf.

`dme.psi` The dyadic model equations right hand sides (or traits) matrix. Present only if dmekeep=TRUE. The name 'psi' refers to the matrix Ψ in equations 12 and 13 of the document dmmOverview.pdf.

`dme.fit` The *fit* object from solving dyadic model equations. Its form depends on the `dmeopt` argument. For dmeopt="qr" (the default) it just contains the QR transform of the DME's. For dmeopt="lm" it contains the object returned by function `lm()` For dmeopt="lmrob" it contains the object returned by function `lmrob()` For dmeopt="pcr" it contains the object returned by function `mvr()` Present only if dmekeepfit=TRUE.

`dme.mean` Means of columns of dyadic model equation matrix

`dme.var` Variances of columns of dyadic model equation matrix

`dme.correl` Correlations between columns of dyadic model equation matrix.

`pcr.loadings` Loadings from principal component regression. Only present when dmeopt="pcr".

`dmeopt` Value of `dmeopt` argument in call to `dmm()` function. Specifies method used to solve DME's, and hence the type of fit object (if one is present).

`siga` Variance component estimates obtained by solving DME's.

`sesiga` Standard errors of variance component estimates.

`vard` Residual (co)variance matrix from solving DME's

`degfd` Degrees of freedom for residual covariance matrix.

`component` Vector of component names from the `component` argument in call to `dmm()` function, with the element "VarP(I)" appended.

`correlation` Estimated genetic or environmental correlation coefficient corresponding to each component

`correlation.variance`
Estimated sampling variance of each correlation coefficient.

`correlation.se` Estimated standard error of each correlation coefficient.

`fraction` Estimated proportion of variance (relative to the total phenotypic (co)variance)) corresponding to each component. For example the proportion corresponding to component "VarG(Ia)" is the usual additive genetic heritability.

`fraction.variance`
Estimated sampling variance of each proportion.

`fraction.se` Estimated standard error of each proportion.

`variance.components`
Variance component estimates (as in *siga*) but with their total which is phenotypic (co)variance appended.

`variance.components.se`
: Standard errors of variance component estimates, including phenotypic (co)variance.

`phenotypic.variance`
: Phenotypic (co)variances as a trait x trait matrix.

`phenotypic.variance.se`
: Standard errors of phenotypic (co)variances as a matrix.

`observed.variance`
: Observed variance, adjusted for fixed effects, in current population. Will differ from phenotypic variance because related animals are correlated. All the estimated components, and their sum (which is phenotypic variance) are estimates of what the components would be in a population of unrelated individuals.

`call`
: The call made to `dmm()` function to generate this object

`gls`
: Another list containing all the above items, but for fixed effects fitted by GLS instead of OLS. Will only be present if gls=TRUE and if the gls iteration converged successfully.

`specific`
: Another list containing those of the above items which are relevant when class-specific (co)varianc components are estimated. Will only be present if the argument `specific.components` is not NULL, that is if at least one class-specific component is fitted.

If option `traitspairwise` is TRUE, the value returned by `dmm()` is an object of class `dmmarray`, which is an array of which each element is an object of class `dmm` representing an analysis for one pair of traits. The rows and columns of the array are named using trait names.

If option `traitsblockwise` is TRUE, the value returned by `dmm()` is an object of class `dmmblockarray`, which is a list of two items named `array` and `blocks`. List item `array` is an array of which each element is an object of class `dmm` representing an analysis for one pair of blocks of traits. The rows and columns of the array are named using block names. List item `blocks` is a list with one element per block, each containig the set of trait names present in the block.

The functions `condense.dmmarray` and `condense.dmmblockarray` are available to facilitate recombining an array of `dmm` objects into a single object of class `dmm` with the variance component and genetic parameter estimates appropriately assembled into multi-trait matrices. For example, one would use these functions to prepare an input file for function `gresponse`.

Note

Two methods of estimating fixed effects are offered by `dmm()` - termed OLS-b and GLS-b. OLS-b is computationally simple and non-iterative and is the default. Use OLS-b for preliminary runs until the set of components to be estimated from the dyadic model equations is settled. Use GLS-b for the final run. OLS-b leads to MINQUE estimates of the variance components and OLS estimates of the fixed effects. GLS-b leads to bias-corrected-ML estimates of the variance components , and GLS estimates of the fixed effects.

The notation for (co)variance components was designed to use ASCII characters only so that it could be usable as dimnames in R. Some examples should make it clear

"VarE(Ia)" variance environmental individual additive

"VarG(Ia)" variance genetic individual additive

"VarG(Ma)" variance genetic maternal additive

"CovG(Ia,Ma)" covariance genetic individual additive x maternal additive

"VarGs(Ia)" variance genetic sexlinked individual additive

For a full coverage of notation see the document dmmOverview.pdf Section 6.

Author(s)

Neville Jackson

References

The document dmmOverview.pdf has a bibliography of literature references.

See Also

Functions `mdf()`, `make.ctable()`, `condense.dmmarray()`, `condense.dmmblockarray()`. Packages nadiv, robustbase, pls

Examples

```
library(dmm)
# Prepare the dataset sheep.df
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A","D"))
# The above code renumbers the pedigree Id's, makes columns "Year","Tb","Sex"
#    into factors,
#    assembles columns "CWW",Diam","Bwt" into a matrix (called 'Ymat')
#    for multivariate processing,
#    and sets up the environmental, additive genetic, and dominance genetic
#    relationship matrices.

# a simple model with individual environmenmtal and
#    additive genetic components (default)
sheep.fit <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
   components=c("VarE(I)","VarG(Ia)"),gls=TRUE)
# view the components and fixed effect coefficients ( 2 traits only)
summary(sheep.fit,traitset=c("Cww","Diam"),gls=TRUE)
# view the genetic parameters
gsummary(sheep.fit,traitset=c("Cww","Diam"))

rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fit)

# Note: sheep.df is a small demo dataset. The results are illustrative,
# but not meaningful.
# for a tutorial and fully documented examples see {\em dmmOverview.pdf}
```

dt8bal.df *A balanced dataset with eight individuals.*

Description

A very small, simple dataset with a balanced design, one fixed effect, and two traits. The design consists of 4 sire families, with 2 offspring per sire, one of each sex.

Usage

```
data(dt8bal.df)
```

Format

A data frame with 20 rows and the following 6 variables.

`Id` Identifier for individuals

`SId` Identifier for sires of individuals

`DId` Identifier for dams of individuals

`Sex` A factor with levels `F` `M`. Sex of individual

`CWW` A numeric vector. Clean wool weight in Kg observed for each individual

`DIA` A numeric vector. Fibre diameter in microns observed for each individual

Details

These data are intended to be used for testing and for demonstrating agreement with analysis of variance estimates in the balanced case.

This dataframe meets the minimal requirements for `dmm()` function; that is its pedigree identifiers are suitably numbered and the base individuals are present. For an univariate analysis it can be utilized directly, without preprocessing by function `mdf()`.

Source

A small subset of real data from an Australian sheep flock.

Examples

```
library(dmm)
data(dt8bal.df)
str(dt8bal.df)
rm(dt8bal.df)
```

gprint *Generic function for printing genetic parameters contained in an object of class* dmm.

Description

Provide a short description of the model fitted and the genetic parameters obtained for an object of class dmm.

Usage

```
gprint(x, ...)
```

Arguments

x	An object of class dmm.
...	See other arguments defined for function gprint.dmm().

Details

This is a short printout without standard errors or confidence limits. It is the analog of print() for a dmm object, but with genetic parameters instead of variance components. For a more extensive printout with standard errors and confidence limits, see function gsummary(). If there are class specific components and genetic parameters, the short printout is repeated for each class.

Value

There is no return value.

Author(s)

Neville Jackson

See Also

Function gprint.dmm().

Examples

```
library(dmm)
# Prepare the dataset sheep.df
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A","D"))

# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
```

```
    components=c("VarE(I)","VarG(Ia)"))
# look at parameters for 2 traits
gprint(sheep.fit1,traitset=c("Cww","Diam"))

rm(sheep.fit1)
rm(sheep.mdf)
rm(sheep.df)
```

`gprint.dmm` *Print method for genetic parameters contained in an object of class* dmm.

Description

Provide a short description of the model fitted and the genetic parameters obtained for an object of class dmm.

Usage

```
## S3 method for class 'dmm'
gprint(x, traitset = "all", gls = F, ...)
```

Arguments

x	An object of class dmm.
traitset	A vector containing the names of the subset of traits for which genetic parameters are to be printed. Default is "all" which means to print parameters for all traits present in object x.
gls	Logical flag: should the parameter estimates by GLS-b method be printed in addition to the parameter estimates by OLS-b method? Default is gls=FALSE. The GLS-b parameters can only be printed if object x contains the attribute gls, that is if x was constructed by a dmm() call with parameter gls=TRUE.
...	Ellipsis argument.

Details

This is a short printout without standard errors or confidence limits. It is the analog of `print()` for a dmm object, but with genetic parameters instead of variance components. For a more extensive printout with standard errors and confidence limits, see function `gsummary()`.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Functions gprint(), gsummary().

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
            sexcode=c("M","F"),relmat=c("E","A","D"))

# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"))
# look at parameters for two traits
gprint(sheep.fit1,traitset=c("Cww","Diam"))
rm(sheep.fit1)
rm(sheep.mdf)
rm(sheep.df)
```

---

gresponse.dmm *Compute response to selection, given phenotypic selection differentials.*

Description

Computes genetic selection differentials, given phenotypic selection differentials and a set of genetic parameters. Effects contributing to response can be any combination of individual additive genetic, individual additive sexlinked genetic, maternal additive genetic, and maternal additive sexlinked genetic. Warning; this function does not currently handle class specific genetic parameters.

Usage

```
## S3 method for class 'dmm'
gresponse(dmmobj, traitset = "all", gls = F,
          psd = list(dp=NULL,dp.sex=NULL,dp.path=NULL),
          effects = "G(Ia)", digits = 3, ...)
```

Arguments

dmmobj	An object of class dmm. Genetic parameters are obtained from this object. Normally this object would be obtained from an output from function dmm(), but there is provision (function make.dmmobj()) for the user to construct a dmm object from external data.

`traitset` A vector containing the names of the subset of traits for which genetic selection differentials are to be computed, and for which phenotypic selection differentials are to be specified. Default is "all" which means all traits present in object `dmmobj`. The order of traits listed here determines the order in all objects returned.

`gls` Logical flag: should the parameter estimates by GLS-b method be used rather than the parameter estimates by OLS-b method? Default is `gls=FALSE`. The GLS-b parameters can only be used if object `dmmobj` contains the attribute `gls`, that is if `dmmobj` was constructed by a `dmm()` call with parameter `gls=TRUE`.

`psd` A list containing the overall (dp), sex-specific (dp.sex), or path-specific (dp.path) phenotypic selection differentials. Only one of these three need be specified. Units for phenotypic selection differentials are the same as units for the traits in the dataframe used to construct `dmmobj`. Units for traits in the phenotypic (co)variance matrix are also the same, albeit squared because they are second moments. The default is a NULL so something must be specified for the `psd` argument or the function will return an error. Phenotypic selection differentials are defined as the difference between the mean of the selected group of individuals and the mean of the whole unselected population, for each trait.

The three `psd` options are specified as follows

dp dp=vector

dp.sex dp.sex=list(he=vector,ho=vector)

dp.path dp.path=list(he.he=vector,ho.he=vector,he.ho=vector,ho.ho=vector)

where 'vector' is always of length equal to the number of traits.

Regardless of which `psd` option is specified the `gresponse()` function always sets up `dp.path` internally and uses this to compute the genetic selection differentials separately for each `path`.

`effects` A vector of character codes specifying the genetic effects contributing to the response computed. There are four options which can be used in any combination:

"G(Ia)" Individual additive genetic effect

"Gs(Ia)" Individual additive sexlinked genetic effect

"G(Ma)" Maternal additive genetic effect

"Gs(Ma)" Maternal additive sexlinked genetic effect

The default is `"G(Ia)"`.

`digits` Number of digits for output. This is returned as part of the return value for use by the S3 print function `print.gresponse.dmm()`.

`...` Ellipsis argument

Details

The `gresponse()` function defines four 'paths' of improvement as follows

he.he Heterogametic sex in the parent to heterogametic sex in the progeny

ho.he Homogametic sex in the parent to heterogametic sex in the progeny

he.ho Heterogametic sex in the parent to homogametic sex in the progeny

ho.ho Homogametic sex in the parent to homogametic sex in the progeny

The response or genetic selection differentials are always calculated separately for each of the four paths, then pooled to give sex specific gsd's, then pooled again to give an overall gsd. This is strictly only needed for responses due to sexlinked effects, but is done for generality.

Clearly if we are computing individual additive genetic responses, the individual additive genetic variance (called "VarG(Ia)") must be available in object `dmmobj`. If computing both individual and maternal additive genetic responses, both the individual and maternal additive genetic variances (called "VarG(Ia)" and "VarG(Ma)") must be available in object `dmmobj`. Their genetic covariances (called "CovG(Ia,Ma)" and "CovG(Ma,Ia)") can optionally be available in object `dmmobj`, if they are not present they are assumed zero. The same applies for sexlinked additive genetic response and maternal sexlinked additive genetic response.

It is advisable to ensure that all parameter matrices are positive definite. In particular the phenotypic covariance matrix must have an inverse.

Value

An object of class `gresponse.dmm`, which is a list containing the following items:

`psdcase`	A character string which is either "overall", "sex", or "path". Describes the type of object used to specify the `psd` argument
`psd`	A list containing one of `dp`, `dp.sex`, `dp.path`, with the others defaulting to NULL, as specifieds in the `psd` argument
`gcov`	The combined genetic covariance matrix. In all cases there will be 4 partitions, one for each of the effects "G(Ia)", "Gs(Ia)", "G(Ma)", and "Gs(Ma)", so the matrix will be of size (4 * l) x (4 * l) where l is number of traits. Effects not specified in argument `effects` will have their corresponding partitions of `gcov` set to zero matrices.
`pcov`	The given phenotypic covariance matrix.
`rmat`	A list containing the R matrices for each path. These incorporate the factor of 0.5 for maternal effects, and the appropriate path factors (0.0, 1.0, or 0.5) for transmission of effects located on the sex chromosome.
`path`	A list containing the genetic selection differentials (gsd) for given psd, the genetic selection differentials (ugsd) for unit psd on each trait, the directional selection gradients (dsg), and the given phenotypic selection differentials (psd). Each of these is given separately for each of the paths (he.he, ho.he, he.ho, ho.ho), where 'he' stands for the heterogametic sex, and 'ho' stands for the homogametic sex. The genetic selection differentials are also given summed across all the effects (gdsum), again separately for each of the paths.
`sex`	A list containing the genetic selection differentials (gsd) achieved by selection of each sex separately (gsd.he. and gsd.ho.), and observed in each sex separately (gsd..he and gsd..ho), as well as summed over effects (gsdsum), again separately for each sex selected, and observed in each sex. The separate phenotypic selection differentials for each sex (psd.he. and psd.ho.)are also given.
`overall`	A list containing the overall genetic selection differentials (gsd), for each effect (gsd..) and summed over effects(gsdsum... The overall phenotypic selection differential (psd..) is also given.

digits	The argument digits specified in the function call
effects	The argument effects specified in the function call
traits	A character vector containing all the specified trait names
call	The function call

Note

There is no provision for computing the effect of non-additive genetic (co)variances on genetic selection differentials. The genetic selection differentials calculated are for a one generation response only. In the case of maternal effects there will be lags in response and the phenotypic response will not match the genetic response. There is no provision for overlapping generations. There is no provision for sex-specific genetic parameters.

Author(s)

Neville Jackson

References

Dickerson,G(1947) Iowa Agricultural Research Station Bulletin No.354 pp489-524

Griffing,B(1966) "Influence of Sex on Selection. III Joint contribution of sex-linked and autosomal genes" Aust. J. Biol. Sci. 19: 775-93

Walsh,B(2009) "Multivariate Selection Response and Estimation of Fitness Surfaces" 2nd Annual NSF short course on Statistical Genetics, Honolulu (13-17 July,2009).

See Also

Functions print.gresponse.dmm(), summary.gresponse.dmm(), make.dmmobj().

Examples

```
library(dmm)
# get some data
data(sheep.df)
# prepare it - only need "E" and "A" relationship matrices
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# estimate genetic parameters - individual and maternal
sheep.fit5 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
              components=c("VarE(I)","VarG(Ia)","VarE(M)","VarG(Ma)",
              "CovG(Ia,Ma)","CovG(Ma,Ia)"))
# compute response using overall psd
sheep.resp <- gresponse(sheep.fit5,psd=list(dp=c(1,1,1)),effects=c("G(Ia)","G(Ma)"))
# look at the response object
summary(sheep.resp)
# cleanup
rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fit5)
```

```
rm(sheep.resp)
```

`gsummary.dmm` *Make summary tables of genetic parameters for a dmm object*

Description

Extracts the genetic parameters from an object of class `dmm`, for the specified set of traits and set of components. Makes tables of parameters ordered either by trait or by component. Tables include parameter estimate, its standard error, and its 95 percent confidence limits. Parameters are grouped intp class-specific classes, if there are any class-specific components fitted.

Usage

```
## S3 method for class 'dmm'
gsummary(dmmobj, traitset = "all", componentset = "all", bytrait = T,
                    gls = F, digits = 3, ...)
```

Arguments

`dmmobj`	An object of class `dmm`. Genetic parameters are obtained from this object.
`traitset`	A vector containing the names of the subset of traits for which tables of genetic parameters are to be constructed. Default is "all" which means all traits present in object `dmmobj`.
`componentset`	A vector containing the names of the subset of (co)variance components for which tables of genetic parmeters are to be constructed. Default is "all" which means all (co)variance components present in object `dmmobj`.
`bytrait`	Logical flag: should the tables of genetic parameters be constructed with trait varying least rapidly from line to line? If TRUE each subtable contains parameters for one trait or traitpair and for all components. If FALSE each subtable contains parameters for one component and for all traits or traitpairs.
`gls`	Logical flag: should the parameter estimates by GLS-b method be tabled in addition to the parameter estimates by OLS-b method? Default is `gls=FALSE`. The GLS-b parameters can only be tabled if object `dmmobj` contains the attribute `gls`, that is if `dmmobj` was constructed by a `dmm()` call with argument `gls=TRUE`.
`digits`	Number of digits for output. This is returned as part of the return value for use by the S3 print function `print.gsummary.dmm()`.
`...`	Ellipsis argument.

Details

This is a long printout with estimates, standard errors and confidence limits, arranged in tables with one estimate per line. It is the analog of `csummary()` for a `dmm` object, but with genetic parameters instead of variance components. In the case of class-specific parameters, the parameters are listed in class groups, so that the proportions of variance sum to unity within each group. In the case where all parameters are non-specific, there is just one class group.

Value

An object of class `gsummary.dmm` which is a list containing the following items:

`ftables`	A list of dataframe objects each containing one subtable of estimates of the proportion of variance attributable to components, along with the appropriate standard errors and confidence limits. Based on OLS-b component estimates.
`rtables`	A list of dataframe objects each containing one subtable of estimates of the correlation coefficient attributable to components, along with the appropriate standard errors and confidence limits. Based on OLS-b component estimates.
`ptables`	A single dataframe object containing estimates of the phenotypic (co)variance. Note that these are based on all components, even if a subset of components is specified in argument `componentset`. Based on OLS-b component estimates.
`gftables`	A list of dataframe objects each containing one subtable of estimates of the proportion of variance attributable to components, along with the appropriate standard errors and confidence limits. Based on GLS-b component estimates. Only present if argument `gls=TRUE`.
`grtables`	A list of dataframe objects each containing one subtable of estimates of the correlation coefficient attributable to components, along with the appropriate standard errors and confidence limits. Based on GLS-b component estimates. Only present if argument `gls=TRUE`.
`gptables`	A single dataframe object containing estimates of the phenotypic (co)variance. Note that these are based on all components, even if a subset of components is specified in argument `componentset`. Based on GLS-b component estimates. Only present if argument `gls=TRUE`.
`traits`	A vector of traitnames as specified in argument `traitset`.
`components`	A vector of component names as specified in argument `componentset`.
`bytrait`	Logical flag: as specified in argument `bytrait`.
`gls`	Logical flag: as specified in argument `gls`.
`digits`	A numeric value, as specified in argument `digits`.
`call`	The function call

Note

There is no provision to constrain the 95 percent confidence limits for parameter estimates. Hence for small samples, these may vary outside the bounds for the parameter.

Author(s)

Nevillle Jackson

See Also

Function `print.gsummary.dmm()`.

Examples

```
# get some data
data(sheep.df)
# prepare it - only need "E" and "A" relationship matrices
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# estimate genetic parameters - individual and maternal
sheep.fit5 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Tb + Sex,
              components=c("VarE(I)","VarG(Ia)","VarE(M)","VarG(Ma)",
              "CovG(Ia,Ma)","CovG(Ma,Ia)"))
# look just at parameter "VarG(Ma)" across all traits
gsummary(sheep.fit5,componentset="VarG(Ma)",bytrait=FALSE)
# look just at trait "Cww"
gsummary(sheep.fit5,traitset="Cww")
# cleanup
rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fit5)
```

---

harv101.df *Harvey dataset*

Description

Real data for average daily gain (Adg) of each of 65 Hereford steers, with age (Age) and initial weight (Weight) as covariates. First used by Walter Harvey in the publication listed below, on page 101 and following pages.

Usage

```
data(harv101.df)
```

Format

A data frame with 139 observations on the following 9 variables.

Id Identifier for individuals

SId Identifier for sires of individuals

DId Identifier for dams of individuals

Line A numeric vector: breeding line for each individual

Agedam A numeric vector: age of dam for each individual

Age A numeric vector: age at weaning for each individual

Weight A numeric vector: initial weight at beginning of test feeding in a feedlot

Adg A numeric vector: average daily gain in weight in the feedlot

Sex A numeric vector: code for Sex of each individual

Details

It has been assumed that all individuals have a unique dam, that is there are no twins or repeat matings. This is not clear in the original presentation. The nonzero relationships in this pedigree are therefore entirely due to individuals having a common sire.

This dataframe is close to meeting the requirements for function `dmm()`. The pedigree Id's are OK, the base animals are present, and there is only one trait to be analysed, so we do not need a traits matrix. However the Line and Agedam need to ba made into factors. We can either fix this by hand, or use function `mdf()`.

Source

Harvey W.R.(1960) "Least Squares Analysis of Data with Unequal Subclass Numbers" United States Department of Agriculture Publication ARS-20-8, July 1960.

Examples

```
library(dmm)
data(harv101.df)
str(harv101.df)
# preprocess, keeping Weight and Adg for use as covariates
# we need the keep=T agrument to preserve the covariates
harv.mdf <- mdf(harv101.df, pedcols=c(1:3), factorcols=c(4,5,9), ycols=3,
            keep=TRUE, sexcode=c(1,2))
str(harv.mdf)
#cleanup
rm(harv101.df)
rm(harv.mdf)
#
# There is a full analysis of this dataset in 'dmmOverview.pdf'.
#
```

`make.countarray` *Count the number of observations in a dataframe or an* `mdf` *object for all traitpairs in the supplied list of traits.*

Description

For some of the traits in a dataframe or an `mdf` object there may be missing observations, coded as `NA`, on some individuals. The function `make.countarray` assembles an array containing observation counts for all pairs of traits from the supplied vector of trait names.

Usage

```
make.countarray(mdf, traits)
```

Arguments

mdf	A dataframe or an object of class mdf. If made with function mdf() the argument keep=TRUE should be used, because the traits must be present as dataframe columns.
traits	A vector of the names of traits given as character strings

Details

This function may be useful if the number of observations varies between traits and one is planning to do a dmm() analysis with either the traitspairwise or traitsblockwise option.

Value

An array object with 2 dimensions, both of size equal to the number of traits in argument traits. Rows and columns are labelled with trait names. Each element is a count of the number of observations not equal to NA for a pair of traits.

Author(s)

Neville Jackson

See Also

Functions dmm(), mdf()

Examples

```
library(dmm)
# prepare the dataset sheep.df
data(sheep.df)
# count the observations
countarray <- make.countarray(sheep.df,c("Cww","Diam","Bwt"))
# lookat the counts
print(countarray)
#cleanup
rm(sheep.df)
rm(countarray)
```

---

make.ctable *Generates a list of vectors containing sets of variance component names*

Description

Internal function used by dmm() to construct vectors containing various subsets of the available variance component names, for use in testing and flow control. Made available at user level because it may be useful in constructing the components argument of dmm().

Usage

```
make.ctable()
```

Details

There are at present 33 variance and cross-effect-covariance components available in `dmm()`. Each of these corresponds to a particular effect in a genetic model. Component names are used throughout `dmm()` to name rows and columns, and hence to label output. For the user, correct specification of component names is vital. The list generated by an internal call to `make.ctable()` is used to check validity, and to manage internal flow control.

Value

A list containing the following items:

`cohortvar`	Vector containing all variance components involving cohort
`cohortcov`	Vector containing all covariance components involving cohort
`cohort`	Vector containing all (co)variance components involving cohort
`evar`	Vector containing all environmental variance comonents
`ecov`	Vector containing all environments covariance components
`e`	Vector containing all environmental (co)variance components
`addgvar`	Vector containing all additive genetic variance components
`domgvar`	Vector containing all dominance genetic variance components
`epiaddgvar`	Vector containing all additive x additive epistatic genetic variance components
`epidomgvar`	Vector containing all epistatic genetic variance components involving dominance
`sexlinaddgvar`	Vector containing all sexlinked additive genetic variance components
`gvar`	Vector containing all genetic variance components
`allvar`	Vector containing all variance components
`addgcov`	Vector containing all additive genetic covariance components
`domgcov`	Vector containing all dominance genetic covariance components
`epiaddgcov`	Vector containing all additive x additive epistatic genetic covariance components
`epidomgcov`	Vector containing all epistatic genetic covariance components involving dominance
`sexlinaddgcov`	Vector containing all sexlinked additive genetic covariance components
`gcov`	Vector containing all genetic covariance components
`g`	Vector containing all genetic (co)variance components
`allcov`	Vector containing all covariance components
`addg`	Vector containing all additive genetic (co)variance components
`domg`	Vector containing all dominance genetic (co)variance components
`epiaddg`	Vector containing all additive x additive epistatic genetic (co)variance components

epidomg	Vector containing all epistatic genetic (co)variance components involving dominance
sexlinaddg	Vector containing all sexlinked additive genetic (co)variance components
indvar	Vector containing all individual variance components
indcov	Vector containing all individual covariance components
ind	Vector containing all individual (co)variance components
matvar	Vector containing all maternal variance components
matcov	Vector containing all maternal covariance components
mat	Vector containing all maternal (co)variance components
all	Vector containing all (co)variance components
allzpre	Vector containing internal codes for type of Z matrix used in constructing column of W
allzpost	Vector containing internal codes for type of Z matrix used in constructing column of W
allrel	Vector containing internal codes for type of relationship matrix used in constructing column of W
cohortzpre1	Vector of special codes for Z matrix for cohort effects
cohortzpost1	Vector of special codes for Z matrix for cohort effects
cohortzpre2	Vector of special codes for Z matrix for cohort effects
cohortzpost2	Vector of special codes for Z matrix for cohort effects
cohortop	Vector of special operator codes for cohort effects

Note

This is an internal function, its definition and its return value may change in future versions.

Component names are defined in the document *dmmOverview.pdf*.

Author(s)

Neville Jackson

See Also

Function `dmm()`.

Examples

```
library(dmm)
# make a ctable
tmp <- make.ctable()
# see its structure
str(tmp)
# look at all additive genetic variances
tmp$addgvar
# tidy up
rm(tmp)
```

make.dmmobj *Construct an object of class* dmm *from user-supplied data*

Description

Construct an object of class dmm containing all attributes needed to run the gresponse() function. The user must supply a phenotypic covariance matrix, and a genetic covariance matrix for each of the components needed by gresponse() Warning; this function does not currently support class specific genetic parameters.

Usage

```
make.dmmobj(p = NULL, components = c("VarG(Ia)"), ...)
```

Arguments

p	A phenotypic covariance matrix. Dimnames for rows and columns must be set to the trait names
components	A character vector specifying names of all of the genetic variance components for which a genetic covariance matrix is to be supplied. If there are genetic covariance components (eg "CovG(Ia,Ma)"), these must be listed after all the genetic variance components
...	A variable number of genetic covariance matrices, one for each of the names listed in the components argument. Each matrix must have dimnames the same as the phenotypic covariance matrix

Details

Only the minimal attributes for use by the gresponse() function are constructed. The remainder are set to NULL

Value

An object of class dmm. Only the following attributes contain data

b	A dummy set of fixed effects
siga	A matrix of genetic variance (and covariance) components set out one component per row and the traitpairs in columns
variance.components	A matrix of genetic variance (and covariance) components with the phenotypic covariance matrix appended
phenotypic.variance	A matrix of phenotypic covariances set out one trait per row and per column

Note

If matrices are supplied which are not positive definite, function `make.dmmobj()` will alter them to the nearest positive definite matrix using routine `neadPD()` from package `Matrix`. No message is given. Check the value returned to see if matrices have been adjusted.

Author(s)

Neville Jackson

See Also

Functions `gresponse()` and `dmm()`.

Examples

```
library(dmm)
p <- matrix(c(3,2,2,4),2,2)
dimnames(p) <- list(c("T1","T2"),c("T1","T2"))
gia <- matrix(c(2,1,1,3),2,2)
dimnames(gia) <- dimnames(p)
myobj <- make.dmmobj(p,components=c("VarG(Ia)"),gia)
myresp <- gresponse(myobj,psd=list(dp=c(0.5,0.1)))
print(myresp)
#cleanup
rm(p)
rm(gia)
rm(myobj)
rm(myresp)
```

---

mdf *Prepare a dataframe for use with dmm function*

Description

The function `mdf()` converts an R dataframe to one which meets the requirements of function `dmm()`, and may optionally append to that dataframe one or more relationship matrices obtained using package `nadiv`. Conversion involves renumbering pedigree Id's, removing duplicates, adding base animals, setting up columns to be fixed factors, putting multivariate traits into a matrix, defining the heterogametic sex, and optionally calling `nadiv` functions to append relationship matrices.

Usage

```
mdf(df, pedcols = c(1:3), factorcols = NULL, ycols = NULL, sexcode = NULL,
    keep = F, relmat = NULL)
```

Arguments

df	A dataframe object with columns labelled: **Id** An identifier for each individual **SId** An identifier for each sire **DId** An identifier for each dam **Sex** A coding for sex of each individual **Fixed effect names** Codings for each fixed effect **Observation names** Numerical values for each trait
pedcols	A vector specifying which columns of df contain the pedigree information (ie Id, SId, and DId). The vector can contain either column numbers, or column names. The dafault is c(1:3).
factorcols	A vector specifying which columns of df contain codes for factors which are to be used as either fixed effects or in defining cohort. The default is NULL.
ycols	A vector specifying which columns of df contain observations which are to become traits in a matrix. The default is NULL. The matrix is always called 'Ymat'.
sexcode	A vector of length 2 specifying the codings used for Sex, with the heterogametic sex code given first position. This should always be specified. The default is NULL. If the Sex column in the dataframe df is a character vector, then sexcode should be a charcter vector. If the Sex column in the dataframe df is an integer vector, then sexcode should be an integer vector. If the Sex column in the dataframe df is a character vector coerced to a factor, then sexcode should be a charcter vector. If the Sex column in the dataframe df is an integer vector coerced to a factor, then sexcode should be an integer vector.
keep	A logical variable. Are columns not specified by pedcols, factorcols, or ycols to be retained in the output object? Default is FALSE - ie unused columns are discarded.
relmat	A vector listing the relationship matrices to be generated and appended to the dataframe thus creating a return object of class mdf. Each relationship matrix has a code letter or name as follows: **"E"** An environmental correlation matrix. At present this produces an identity matrix - ie no environmental correlation effects. Must always be included. **"A"** Additive genetic relationship matrix. **"D"** Dominance relationship matrix. **"Dsim"** Dominance relationship matrix by the simulation method (see nadiv). **"AA"** Additive x additive epistatic relationship matrix. **"AD"** Additive x dominance epistatic relationship matrix. **"DD"** Dominance x dominance relationship matrix. **"S"** Sex linked additive genetic relationship matrix with no global dosage compensation ('ngdc' option see nadiv) **"S.hori"** Sex linked additive genetic relationship matrix with 'hori' dosage compensation model (see nadiv) **"S.hedo"** Sex linked additive genetic relationship matrix with 'hedo' dosage compensation model (see nadiv)

"S.hoha" Sex linked additive genetic relationship matrix with 'hoha' dosage compensation model (see `nadiv`)

"S.hopi" Sex linked additive genetic relationship matrix with 'hopi' dosage compensation model (see `nadiv`)

Default is NULL - ie no relationship matrices constructed.

Details

If planning to use numerical observations as covariates in the fixed effects model under `dmm()` use argument `keep=TRUE`, so that the covariate columns are retained in the returned dataframe object.

The following actions are performed by `mdf()`:

- remove any Id's which are NA or duplicate (including first duplicate)
- add SId's which do not match any Id as base Id's
- add DId's which do not match any Id as base Id's
- renumber all Id's
- retain original Id's as row names
- if `keep=TRUE` retain unused columns of dataframe
- if `keep=FALSE` do not retain unused columns of dataframe
- always retain Id, SId, DId, and factors
- Sex should be one of the factors
- transform Sex codes to NA if not in argument `sexcode[]`
- take first entry in `sexcode[]` as the heterogametic sex
- make columns in `factorcols` into factors
- make columns in `ycols` into a matrix of traits called 'Ymat'
- if `relmat` argument is present, compute the relationship matrices specified and make a returned list object `mdf` containing the modified dataframe as `mdf$df` and the relationship matrices as `mdf$rel`
- if `relmat` argument is not present simply return the modified dataframe

Value

The return object is of class `mdf` if relationship matrices are requested, and is of class `dataframe` if relationship matrices are not requested.

An object of class `mdf` is a list containing the following items:

df A dataframe conforming to the requirements of function `dmm()`

rel A list of relationship matrices

An object of class `dataframe` as returned by function `mdf()` is a dataframe conforming to the requirements of function `dmm()`

Note

Individuals which appear in the SId or DId columns, but not in the Id column are assumed to be 'base individuals', ie they have unknown sire and dam. They will be given an Id and added to the dataframe, but their SId and DId and all data except for Sex coding will be set to NA, so they will be assumed unrelated and will not contribute data. It is important that 'base individuals' be present for relationship matrices to be calculated correctly.

Author(s)

Neville Jackson

See Also

Functions `dmm()`, `pedrenum()`. Package `nadiv`

Examples

```
library(dmm)

# prepare a multi-trait dataset from sheep.df
data(sheep.df)
# look at its structure
str(sheep.df)
# needs some work - Id, SId, DId are alphanumeric
#                 - Year is numeric and we want it as a factor
#                 - there are 3 traits (Cww,Diam,Bwt) to put into a trait matrix
sheep.mdf1 <- mdf(sheep.df,pedcols=c(1:3), factorcols=c(4:6), ycols=c(7:9),
            sexcode=c("M","F"))
# note the screen messages - it also had to add 2 base Id's for 2 of the dams
str(sheep.mdf1)
# so it returned a dataframe object with 44 observations
# and one of the columns is a matrix called 'Ymat'

# prepare a dataset requiring relationship matrices
sheep.mdf2 <- mdf(sheep.df,pedcols=c(1:3), factorcols=c(4:6), ycols=c(7:9),
            sexcode=c("M","F"),relmat=c("E","A"))
# note the screen messages - it now makes an object of class mdf
str(sheep.mdf2)
# so it returned a list object with 2 items
#    df - the dataframe
#   rel - a list of relationship matrices ( note those not requested are NULL)
#
```

`merino.df` *Australian Merino sheep research dataset*

Description

A set of real data from an Australian Merino sheep research flock with a multi-generation pedigree, eight fixed effects, and 11 traits related to wool production.

Usage

```
data(merino.df)
```

Format

A data frame with 4449 observations on the following 22 variables.

`Id` Identifier for individuals

`SId` Identifier for sires of individuals

`DId` Identifier for dams of individuals

`Sex` A factor with levels `M` (male) `F` (female)

`Yearbi` A factor: year of birth of each individual

`YearSbi` A factor: year of birth of each individual's sire

`YearDbi` A factor: year of birth of each individual's dam

`Mob` A factor: each individual was reared in one of two grazing environments coded `1` and `2`

`Agem` A factor: each individual was measured at one of two ages coded `3` (12 months) and `9` (15 months)

`Birwt` A numeric vector: weight at birth in Kg

`Weanwt` A numeric vector: weight at weaning in Kg

`Birls` A factor: litter size at birth coded as `1` and `2`

`Weanls` A factor: litter size at weaning coded as `1` and `2`

`Crimp` A numeric vector: staple crimp frequency in crimps per 2.5cm

`Densty` A numeric vector: density of wool fibres on the skin surface in fibres per square mm

`Diamtr` A numeric vector: average fibre diameter in microns

`Yield` A numeric vector: wool yield as a percentage. The amount of clean wool as a percentage of the greasy weight of fleece

`Bodywt` A numeric vector: weight in Kg at the time of wool measurement, that being either 12 or 15 months, see item `Agem`

`Wrinkl` A numeric vector: a visual score for skin wrinkle

`Length` A numeric vector: staple length in cm

`Flcwt` A numeric vector: greasy fleece weight in Kg

`Woolwt` A numeric vector: clean wool weight in Kg

Details

These data are intended to show the utility of `dmm()` for analysis of a multi-trait dataset with all the real world complications. The dataframe has identifiers which are NA or duplicate or alphanumeric, some base animals are missing, some traits and factors have missing values, and the eleven traits need to be in a matrix for multivariate analysis.

Source

These data were collected over the period 1950 to 1970 by CSIRO under the direction of Dr Helen Newton Turner. The dataset was recovered from the author's research files, and are used with permission from CSIRO.

References

These data form part of the set which were used to estimate genetic parameters in the publication: Brown,G.H. and Turner,Helen Newton(1968)Aust.J.Agric.Res.19:303-22

For a coverage of Australian Merino sheep research see : Turner,Helen Newton and Young,S.S.Y(1969) Quantitative Genetics in Sheep Breeding. Macmillan, Melbourne, 1969.

Examples

```
library(dmm)
data(merino.df)
str(merino.df)
rm(merino.df)
#
# there is a full analysis of this dataset in 'dmmOverview.pdf'.
#
```

---

pedcheck *Checks that the Id, SId, and DId columns of a dataframe are valid for function* `dmm()`.

Description

Checks that Id's form an arithmetic sequence, and that every SId and DId appears as an Id.This ensures that relationship matrix construction can proceed without error. Also checks each SId is a male, and each DId is a female

Usage

```
pedcheck(df)
```

Arguments

`df`	A dataframe containing the columns Id, SId, and DId as required to include pedigree information.

Details

Function `pedcheck()` would normally be used before function `mdf()`, to indicate whether `mdf()` is needed. If there are base animals missing from the dataframe, it will report errors. It can be used after `mdf()` just to check for errors in the Sex of individuals. Errors in the Sex of individuals can affect calculation of sexlinked relationship matrices.

Both the inline code in function `dmm()` and the routines in package `nadiv` require Id's in a numerical sequence and base animals present, for correct relationship matrix calculations.

Value

Returns zero if the dataframe passes check tests. Returns number of message(s) if the dataframe fails check tests, and prints the messages.

Note

Function `pedcheck()` is an internal function called by function `dmm()`. It is made available because it may be useful for preliminary checking.

Author(s)

Neville Jackson

Examples

```
library(dmm)
data(dt8bal.df)
pedcheck(dt8bal.df)
# this one returns
#[1] 0
# which is a pass
## Not run:
data(sheep.df)
pedcheck(sheep.df)
# this one returns
#Id's must start at 1:
#Id's must be an arithmetic sequence:
#Id's must be unique:
#DId's must occur as an Id in the dataframe:
#All DId's must be female:
#[1] 5
# which is a fail

## End(Not run)
rm(dt8bal.df)
```

`pedrenum` *Renumbers the Id, SId, and DId columns of a dataframe*

Description

Converts the identifiers in column Id of a dataframe into numeric codes which form a sequence from 1 to n with unit increments and no duplicates. Converts the identifiers in columns SId and DId to correspond.

Usage

```
pedrenum(df)
```

Arguments

df	A dataframe containing columns named Id, SId, and DId as required to include pedigree information

Details

It is assumed that any identifiers in columns SId or DId also appear in column Id. If not use function mdf() instead of function pedrenum(). Function mdf() actually calls function pedrenum(), but ensures that the above requirement is met first.

Value

A dataframe containing the renumbered Id, SId, and DId columns, as well as any other columns present in argument df

Note

Function pedrenum() is an internal function called by function mdf(). It is made available because it may be useful in cases where the complex dataframe manipulations performed by function mdf() are not required.

Author(s)

Neville Jackson

See Also

Function mdf()

Examples

```
library(dmm)
data(dt8bal.df)
# note these data do not need renumbering, but will use as a demo
tmprenum <- pedrenum(dt8bal.df)
# In this case all that happens is that SId, which was a factor in dt8bal.df, is
# converted back to int. The numeric codes are unaltered.
rm(dt8bal.df)
rm(tmprenum)
```

plot.dmm *Plot residuals from fitting dyadic model*

Description

Plots dyadic residuals as five separate plots showing histogram of residuals, qqnorm plot of residuals, fitted values against residuals, dyadic covariances against residuals, and dyadic covariances against fittes values. Multi trait case shows all trait pairs on each plot.

Usage

```
## S3 method for class 'dmm'
plot(x, traitset = "all", gls = F, ...)
```

Arguments

x	An object of class dmm. This will be a 'fit' object for the dyadic model equations. It should contain attributes dme.fit and dme.psi obtained by calling dmm() with arguments dmekeep=TRUE and dmekeepfit=TRUE respectively.
traitset	Either a character vector specifying trait names to be plotted, or the default valuse which is "all" meaning plot all traits present in object x.
gls	A logical flag. Should the plot be of dyadic residuals given OLS-b fixed effects, or of dyadic residuals given GLS-b fixed effects. Default is gls=FALSE.
...	Other arguments passed to plotting functions.

Details

In plots with gls=FALSE there will be N^2 residuals, where N is the number of individuals with data. In plots with gls=TRUE there will be $N^2 * L^2$ residuals, where L is the number of traits. This is because the GLS-b fit is always multivariate, whereas the OLS-b fit is multi-trait, just like a multiple regression with multi-trait response.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Function print.dmm() .

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"),
    dmekeep=TRUE,dmekeepfit=TRUE)
# plot dyadic model residuals for all traits
plot(sheep.fit1)
#cleanup
rm(sheep.fit1)
rm(sheep.mdf)
rm(sheep.df)
```

---

print.csummary.dmm *Print method for object of class* csummary.dmm.

Description

Formats each attribute of a csummary.dmm object for printing, adding appropriate headings.

Usage

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

Arguments

x	An object of class csummary.dmm.
...	Ellipsis argument

Details

Each table in an object of class csummary.dmm is a dataframe. This method uses the default dataframe print method to format each table for printing, passing the digits attribute from the csummary.dmm object to the print call.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Function csummary.dmm().

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# make a simple fit object - OLS-b only
sheep.fits <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
          components="VarE(I)",specific.components=list(Sex="VarG(Ia)"))
# compute a 'csummary.dmm' object, use all the defaults
sheep.csum <- csummary(sheep.fits)
# print the summary of genetic parameters
print(sheep.csum)
## Not run:
# can do the same thing without saving response object
csummary(sheep.fits)
# so this is the default print method for an object of class 'csummary.dmm'

## End(Not run)
#cleanup
rm(sheep.fits)
rm(sheep.csum)
rm(sheep.mdf)
rm(sheep.df)
```

---

print.dmm *Print method for a* dmm() *fitted model object.*

Description

Provide a short description of the model fitted and the fixed effects and (co)variance component estimates obtained for an object of class dmm.

Usage

```
## S3 method for class 'dmm'
print(x, traitset = "all", gls = F, ...)
```

Arguments

x	An object of class dmm.
traitset	A vector containing the names of the subset of traits for which fixed effects and (co)variance components are to be printed. Default is "all" which means to print estimates for all traits present in object x.

gls	Logical flag: should the fixed effects and (co)variance component estimates by GLS-b method be printed in addition to the fixed effects and (co)variance component estimates by OLS-b method? Default is gls=FALSE. The GLS-b fixed effects and (co)variance component estimates can only be printed if object x contains the attribute gls, that is if x was constructed by a dmm() call with argument gls=TRUE.
...	Ellipsis argument.

Details

This is a short printout without standard errors or confidence limits. For a more extensive printout with standard errors and confidence limits, see function summary.dmm(). The printout includes fixed effects, variance component estimates, and correlations among columns of the W matrix of the dyadic model equations.

Value

There is no return value. Function is used for its side effects.

Note

For a similar short printout, but with genetic parameters instead on (co)variance components, see function gprint.dmm().

Author(s)

Neville Jackson

See Also

Functions summary.dmm() and gprint.dmm().

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A","D"))
# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"))
# look at model plus fixed effects and components for all traits
print(sheep.fit1)
## Not run:
# can do the same thing without saving fit object
dmm(sheep.mdf, Ymat ~ 1 + Year + Tb + Sex,
    components=c("VarE(I)","VarG(Ia)"))
# so this is the default print method for an object of class 'dmm'

## End(Not run)
```

```
#cleanup
rm(sheep.fit1)
rm(sheep.mdf)
rm(sheep.df)
```

`print.gresponse.dmm` *Print method for object of class* `gresponse.dmm`.

Description

Prints overall responses only.

Usage

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

Arguments

`x`	An object of class `gresponse.dmm`.
`...`	Ellipsis argumnet.

Details

This is a simple printout of overall response estimates. There are no standard errors or confidence limits for response estimates.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Functions `gresponse.dmm()`, `summary.gresponse.dmm()`.

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"))
```

```
# compute some response estimates, use all the defaults
sheep.resp <- gresponse(sheep.fit1,psd=list(dp=c(1,1,1)))
# print these
print(sheep.resp)
## Not run:
# can do the same thing without saving response object
gresponse(sheep.fit1,psd=list(dp=c(1,1,1)))
# so this is the default print method for an object of class 'gresponse.dmm'

## End(Not run)
#cleanup
rm(sheep.fit1)
rm(sheep.resp)
rm(sheep.mdf)
rm(sheep.df)
```

`print.gsummary.dmm` *Print method for object of class* `gsummary.dmm`.

Description

Formats each attribute of a `gsummary.dmm` object for printing, adding appropriate headings.

Usage

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

Arguments

`x`	An object of class `gsummary.dmm`.
`...`	Ellipsis argument

Details

Each table in an object of class `gsummary.dmm` is a dataframe. This method uses the default dataframe print method to format each table for printing, passing the `digits` attribute from the `gsummary.dmm` object to the print call.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Function `gsummary.dmm()`.

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
   components=c("VarE(I)","VarG(Ia)"))
# compute a 'gsummary.dmm' object, use all the defaults
sheep.gsum <- gsummary(sheep.fit1)
# print the summary of genetic parameters
print(sheep.gsum)
## Not run:
# can do the same thing without saving response object
gsummary(sheep.fit1)
# so this is the default print method for an object of class 'gsummary.dmm'

## End(Not run)
#cleanup
rm(sheep.fit1)
rm(sheep.gsum)
rm(sheep.mdf)
rm(sheep.df)
```

`print.summary.dmm` *Print method for an object of class* `summary.dmm`.

Description

Formats each attribute of a `summary.dmm` object for printing, adding appropriate headings.

Usage

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

Arguments

`x`	An object of class `summary.dmm`.
`...`	Ellipsis argument.

Details

Each table in an object of class `summary.dmm` is a dataframe. This method uses the default dataframe print method to format each table for printing, passing the `digits` attribute from the `summary.dmm` object to the print call.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Function `summary.dmm()`.

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
    components=c("VarE(I)","VarG(Ia)"))
# compute a 'summary.dmm' object, use all the defaults
sheep.sum <- summary(sheep.fit1)
# print the summary of genetic parameters
print(sheep.sum)
## Not run:
# can do the same thing without saving response object
summary(sheep.fit1)
# so this is the default print method for an object of class 'summary.dmm'

## End(Not run)
#cleanup
rm(sheep.fit1)
rm(sheep.sum)
rm(sheep.mdf)
rm(sheep.df)
```

---

quercus.df *Quercus example dataset*

Description

Example dataset from the program package 'QUERCUS', developed by Ruth G. Shaw and Frank H. Shaw. Known to 'quercus' as the 'demo2' dataset.

Usage

```
data(quercus.df)
```

Format

A data frame with 180 observations on the following 6 variables.

`Id` Identifier for individuals

`SId` Identifier for sires of individuals

`DId` Identifier for dams of individuals

`Sex` A numeric vector: code for Sex of each individual

`Trait1` A numeric vector: an observation called Trait1

`Trait2` A numeric vector: an observation called Trait2

Details

This is a simulated dataset suitable for a 3-component analysis, the components being environmental variance, additive genetic variance, and dominance genetic variance. It needs pre-processing with function `mdf()` to add base animals, to combine the two traits into a matrix, and to calculate additive and dominance relationship matrices.

Source

https://cbs.umn.edu/eeb/about-eeb/helpful-links/quercus-quantitative-genetics-software

Examples

```
library(dmm)
data(quercus.df)
str(quercus.df)
# preprocess
quercus.mdf <- mdf(quercus.df,pedcols=c(1:3),factorcols=4,ycols=c(5:6),
               relmat=c("E","A","D"),sexcode=c(1,2))
str(quercus.mdf)
# cleanup
rm(quercus.df)
rm(quercus.mdf)
#
# there is a full analysis of this dataset in 'dmmOverview.pdf'.
#
```

---

sheep.df *Demonstration sheep dataset*

Description

A small dataset with an unbalanced design, three fixed effects one of which can be interpreted as a cohort, three traits, and a pedigree which permits most available variance components to be fitted. Deliberately set up to be useful for a variety of demonstrations.

Usage

```
data(sheep.df)
```

Format

A data frame with 42 observations on the following 9 variables.

`Id` Identifier for individuals

`SId` Identifier for sires of individuals

`DId` Identifier for dams of individuals

`Year` A numeric vector: year of birth of each individual

`Tb` A factor with levels `S` (born as a single lamb) `T` (born as a twin lamb)

`Sex` A factor with levels `M` (male) `F` (female)

`Cww` A numeric vector. Clean wool weight in Kg observed for eac h individual

`Diam` A numeric vector. Fibre diameter in microns observed for e ach individual

`Bwt` A numeric vector. Body weight in Kg observed for each individual

Details

These data are intended for demonstration, and are extensively used in examples in the `dmm` package help files.

This dataframe does not meet the minimum requirements for function `dmm()`. The identifiers are alphanumeric, some base animals are missing, and the three traits need to be in a matrix for multivariate analysis. It requires preprocessing by function `mdf()`.

Source

A small subset of real data from an Australian sheep flock. Not the whole flock, and not a random sample.

Examples

```
library(dmm)
data(sheep.df)
str(sheep.df)
#do some preprocessing
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# The above code renumbers the pedigree Id's, makes columns "Year","Tb","Sex"
#    into factors,
#    assembles columns "CWW",Diam","Bwt" into a matrix (called 'Ymat')
#    for multivariate processing,
#    and sets up the environmental, and additive genetic
#    relationship matrices.
str(sheep.mdf)
#cleanup
rm(sheep.df)
rm(sheep.mdf)
```

summary.dmm *Make summary tables of (co)variance component estimates and fixed effect estimates for a dmm object.*

Description

Extracts the (co)variance component and fixed effect estimates from an object of class dmm, for the specified set of traits and set of components. Makes tables of component estimates ordered either by trait or by component. Tables include component estimate, its standard error, and its 95 percent confidence limits. If there are class specific components these appear with appropriate labels in the list of components.

Usage

```
## S3 method for class 'dmm'
summary(object, traitset = "all", componentset = "all", bytrait = T,
            gls = F, digits = 3, ...)
```

Arguments

object	An object of class dmm. (Co)variance component estimates are obtained from this object.
traitset	A vector containing the names of the subset of traits for which tables of (co)variance component estimates are to be constructed. Default is "all" which means all traits present in object object.
componentset	A vector containing the names of the subset of (co)variance components for which tables are to be constructed. Default is "all" which means all (co)variance components present in object object.
bytrait	Logical flag: should the tables of (co)variance component estimates be constructed with trait varying least rapidly from line to line? If TRUE each subtable contains component estimates for one trait or traitpair and for all components in argument componentset. If FALSE each subtable contains component estimates for one component and for all traits or traitpairs.
gls	Logical flag: should the (co)variance component estimates by GLS-b method be tabled in addition to the (co)variance component estimates by OLS-b method? Default is gls=FALSE. The GLS-b (co)variance component estimates can only be tabled if object object contains the attribute gls, that is if object was constructed by a dmm() call with argument gls=TRUE.
digits	Number of digits for output. This is returned as part of the return value for us e by the S3 print function print.summary.dmm().
...	Ellipsis argument.

Details

This is a long printout with estimates, standard errors and confidence limits, arranged in tables with one estimate per line. For a short printout see function `print.dmm()`. In the case of class-specific components, the components which are class-specific are labelled with their class-codes prepended to the variance component name, and components for all classes are included, so the listed components do not sum to phenotypic variance. To list the components in class groups, so that they sum to the class phenotypic variance use function `csummary()`.

Value

An object of class `gsummary.dmm` which is a list containing the following items:

`btables`	A list of dataframe objects each containing one subtable of estimates of the fixed effects, along with the appropriate standard errors and confidence limits. Based on OLS-b fixed effect estimates.
`ctables`	A list of dataframe objects each containing one subtable of estimates of the (co)variance components, along with the appropriate standard errors and confidence limits. Based on OLS-b component estimates.
`gbtables`	A list of dataframe objects each containing one subtable of estimates of the fixed effects, along with the appropriate standard errors and confidence limits. Based on GLS-b fixed effect estimates.
`gctables`	A list of dataframe objects each containing one subtable of estimates of the (co)variance components, along with the appropriate standard errors and confidence limits. Based on GLS-b component estimates. Only present if argument `gls=TRUE`.
`traits`	A vector of traitnames as specified in argument `traitset`.
`components`	A vector of component names as specified in argument `componentset`.
`bytrait`	Logical flag: as specified in argument `bytrait`.
`gls`	Logical flag: as specified in argument `gls`.
`digits`	A numeric value, as specified in argument `digits`.
`call`	The function call

Note

There is no provision to constrain the 95 percent confidence limits for component estimates. Hence for small samples, these may vary outside the bounds for the component, that is for components which are variances, they may be negative. Fixed effects are not bounded. Use `summary()` if you want to see the components as estimated. Use `csummrry()` if you want to see the components summing to phenotypic (co)variance, or sorted into class-specific groups.

Author(s)

Neville Jackson

See Also

Function `print.summary.dmm()`.

Examples

```
# get some data
data(sheep.df)
# prepare it - only need "E" and "A" relationship matrices
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# estimate (co)variance components  - individual and maternal
sheep.fit5 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
              components=c("VarE(I)","VarG(Ia)","VarE(M)","VarG(Ma)",
              "CovG(Ia,Ma)","CovG(Ma,Ia)"))
# look just at component "VarG(Ma)" across all traits
summary(sheep.fit5,componentset="VarG(Ma)",bytrait=FALSE)
# look just at trait "Cww"
summary(sheep.fit5,traitset="Cww")
# cleanup
rm(sheep.df)
rm(sheep.mdf)
rm(sheep.fit5)
```

`summary.gresponse.dmm` *Summary method for object of class* `gresponse.dmm`.

Description

Summarizes path specific, sex specific, and overall responses in a `gresponse.dmm` object, adding appropriate headings.

Usage

```
## S3 method for class 'gresponse.dmm'
summary(object, ...)
```

Arguments

`object`	An object of class `gresponse.dmm`.
`...`	Ellipsis argumnet.

Details

This is a summary of response estimates and the parameters used to calculate them. It includes path specific, sex specific and overall responses. There are no standard errors or confidence limits for response estimates.

Value

There is no return value. Function is used for its side effects.

Author(s)

Neville Jackson

See Also

Functions `gresponse.dmm()`, `print.gresponse.dmm()`.

Examples

```
library(dmm)
data(sheep.df)
sheep.mdf <- mdf(sheep.df,pedcols=c(1:3),factorcols=c(4:6),ycols=c(7:9),
             sexcode=c("M","F"),relmat=c("E","A"))
# make a simple fit object - OLS-b only
sheep.fit1 <- dmm(sheep.mdf, Ymat ~ 1 + Year + Sex,
   components=c("VarE(I)","VarG(Ia)"))
# compute some response estimates, use all the defaults
sheep.resp <- gresponse(sheep.fit1,psd=list(dp=c(1,1,1)))
# summarize these
summary(sheep.resp)
#cleanup
rm(sheep.fit1)
rm(sheep.resp)
rm(sheep.mdf)
rm(sheep.df)
```

---

tstmo1.df *Dfreml example dataset*

Description

Example dataset from the program 'DFREML', developed by Karin Meyer. Known to 'DFREML' as the 'Example 1' dataset. A univariate dataset with 282 individuals in a 2 generation pedigree with full-sib families.

Usage

```
data(tstmo1.df)
```

Format

A data frame with 282 observations on the following 6 variables.

`Id` Identifier for individuals

`SId` Identifier for sires of individuals

`DId` Identifier for dams of individuals

`Sex` A factor: Sex of each individual

`Gen` A numeric vector: generation number of each individual

`Weight` A numeric vector: weight in ? of each individual

Details

Karin Meyer gives the following description of these data:

" .... The test data given is that used by Meyer(1989) to illustrate univariate REML estimation via a derivtive-free algorithm. They are simulated records for a trait with a phenotypic variance of 100, direct heritability of 0.40, maternal heritability of 0.20, maternal-direct covariance (divided by 100) of -0.05, and a "c-squared" effect of 0.15. Data were generated for 2 generations of animals with a heirarchical full-sib family structure, yielding a total of 282 records and 306 animals in the analysis with generations as the only fixed effect. "

Source

DFREML Version 3.0 User Notes. Karin Meyer. September 9,1998.

These data were distributed with the DFREML program. Note DFREML is not currently available.

References

Meyer,K.(1989) Restricted Maximum Likelihood to estimate variance components for animal models with several random effects using a derivative-free algorithm. Genet. Select. Evol. 21:317-340

Examples

```
library(dmm)
data(tstmo1.df)
str(tstmo1.df)
rm(tstmo1.df)
#
# There is a full analysis of this dataset in 'dmmOverview.pdf'.
#
```

---

unfactor *Convert a vector from factor to numeric*

Description

Convert a vector (such as a dataframe column) from factor to numeric. Non-numeric values will coerce to NA.

Usage

```
unfactor(x)
```

Arguments

x	A vector of type factor. Typically this would be one column of a dataframe.

Details

This function may be useful when preparing a dataframe for dmm(). It is a common problem for dataframe columns to be automatically made type factor when constructing the dataframe with functions such as read.table, due to the presence of a small number of non-numeric values. Dataframe columns used as traits or as covariates should not be of type factor.

Value

A vector of numeric values is returned.

Author(s)

Neville Jackson

See Also

Functions dmm(), read.table()

Examples

```
library(dmm)
tmp <- as.factor(c(1,2,3))
str(tmp)
utmp <- unfactor(tmp)
str(utmp)
rm(tmp)
rm(utmp)
```

---

warcolak.convert *Convert* warcolak *data file to format required for a dataframe for* dmm() *or* mdf().

Description

A simple function to relabel the columns of the warcolak dataset from package nadiv, to comply with the requirements of dmm() or mdf().

Usage

```
warcolak.convert(w)
```

Arguments

w	An object produced by the call data(warcolak) from package nadiv.

Details

The warcolak dataset has columns named slightly differently from what dmm() requires, but is otherwise compatable. We use this function to do a simple conversion before using warcolak as test data for dmm().

Value

A dataframe containing the following columns:

"Id" Individual identifier

"SId" Sire identifier

"DId" Dam identifier

"Sex" Coding for sex

"Trait1" First trait phenotypic value

"Trait2" Second trait phenotypic value

"t1_a" First trait additive genetic effect

"t2_a" Second trait additive genetic effect

"t2_s" Second trait additive genetic sexlinked effect

"t1_d" First trait dominance genetic effect

"t2_d" Second trait dominance genetic effect

"t1_r" First trait environmental effect

"t2_r" Second trait environmental effect

Author(s)

Neville Jackson

See Also

Functions dmm(), mdf(). Package nadiv.

Examples

```
#library(dmm)
#data(warcolak)
#warcolak.df <- warcolak.convert(warcolak)
#str(warcolak.df)
#rm(warcolak.df)
#rm(warcolak)
```

Index