Package 'miceadds'

November 4, 2015

Type Package

Title Some Additional Multiple Imputation Functions, Especially for 'mice'

Version 1.4-0

Date 2015-11-04

Author Alexander Robitzsch [aut, cre]

Maintainer Alexander Robitzsch <robitzsch@ipn.uni-kiel.de>

Description Contains some auxiliary functions for multiple imputation which complements existing functionality in R.
In addition to some utility functions, main features include plausible value imputation, multilevel imputation functions, imputation using partial least squares (PLS) for high dimensional predictors, nested multiple imputation, and two-way imputation.

Depends R (>= 2.15.0), mice

Imports methods, stats, graphics, utils, Rcpp, TAM, lme4, inline, car, foreign, sjmisc, multiwayvcov, grouped, MBESS, bayesm, sirt, pls, mitools, MASS, mvtnorm

LinkingTo Rcpp, RcppArmadillo

Suggests Amelia, Zelig, jomo, haven, BIFIEsurvey, mitml, pan

License GPL (>= 2)

URL https://sites.google.com/site/alexanderrobitzsch/software

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-11-04 19:04:40

R topics documented:

`miceadds-package` *Some Additional Multiple Imputation Functions, Especially for* **mice**

Description

Contains some auxiliary functions for multiple imputation which complements existing functionality in R. In addition to some utility functions, main features include plausible value imputation, multilevel imputation functions, imputation using partial least squares (PLS) for high dimensional predictors, nested multiple imputation, and two-way imputation.

Details

`DESCRIPTION` **miceadds** package

Package:	**miceadds**
Type:	Package
Version:	1.4
Publication Year:	2015
License:	GPL (>= 2)
URL:	https://sites.google.com/site/alexanderrobitzsch/software

- In addition to the usual `mice` imputation function which employs parallel chains, the function `mice.1chain` does multiple imputation from a single chain.
- Nested multiple imputation can be conducted with `mice.nmi`.
- Imputation based on partial least squares regression is implemented in `mice.impute.2l.pls`.
- Unidimensional plausible value imputation for latent variables (or variables with measurement error) in the **mice** sequential imputation framework can be applied by using the method `mice.impute.2l.plausible.values`.
- Imputations for questionnaire items can be accomplished by two-way imputation (`tw.imputation`).

- The **miceadds** package also includes some functions R utility functions (e.g. `write.pspp`, `ma.scale2`).

Author(s)

Alexander Robitzsch
IPN - Leibniz Institute for Science and Mathematics Education at Kiel University

Maintainer: Alexander Robitzsch <robitzsch@ipn.uni-kiel.de>

URL: https://sites.google.com/site/alexanderrobitzsch/

See Also

See other R packages for conducting multiple imputation: **mice**, **Amelia**, **pan**, **mi**, **norm**, **BaBooN**, **VIM**, ...

Some links to internet sites related to missing data:

http://missingdata.lshtm.ac.uk/
http://www.stefvanbuuren.nl/mi/
http://www.bristol.ac.uk/cmm/software/realcom/

Examples

```
##
##   ::''''''''''''''''''''''''''''''''''''::
##   :: miceadds 0.11-69 (2013-12-01)   ::
##   ::''''''''''''''''''''''''''''''''''''::
##
##  ----------------------- mice at work ---------------------------------
##
##                          (\-.
##                          / _`> .---------.
##                  _)     / _)=  |'-------'|
##                (       / _/    |O   O   o|
##                 `-.__(___)_    | o O . o |
##                                `---------'
##
##                                        oo__
##                                       <;___)------
##                                    oo__   " "
##                                   <;___)------     oo__
##                                     " "           <;___)------
##                                                     " "
```

`complete.miceadds` *Creates Imputed Dataset from a* `mids.nmi` *or* `mids.1chain` *Object*

Description

Creates imputed dataset from a `mids.nmi` or `mids.1chain` object.

Usage

```
complete.mids.nmi( x , action = c(1,1) )

complete.mids.1chain( x , action = 1 )
```

Arguments

`x`	Object of class `mids.nmi` (for `complete.mids.nmi`) or `mids.1chain` (for `complete.mids.1chain`)
`action`	A vector of length two indicating to indices of between and within imputed dataset for for `complete.mids.nmi` and an integer for the index of imputed dataset for `complete.mids.1chain`.

Author(s)

Alexander Robitzsch

See Also

See also the corresponding `mice::complete` function.

Imputation methods: `mice.nmi`, `mice.1chain`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Nested multiple imputation and dataset extraction for TIMSS data
#############################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2

# remove first four variables
M <- length(datlist)
for (ll in 1:M){
    datlist[[ll]] <- datlist[[ll]][ , -c(1:4) ]
                }

#***************
# (1) nested multiple imputation using mice
```

```
imp1 <- mice.nmi( datlist ,  m=4 , maxit=3 )
summary(imp1)

#***************
# (2) nested multiple imputation using mice.1chain
imp2 <- mice.nmi( datlist , Nimp=4 , burnin=10 ,iter=22, type="mice.1chain")
summary(imp2)

#**************
# extract dataset for third orginal dataset the second within imputation
dat32a <- complete.mids.nmi( imp1 , action = c(3,2) )
dat32b <- complete.mids.nmi( imp2 , action = c(3,2) )

##############################################################################
# EXAMPLE 2: Imputation from one chain and extracting dataset for nhanes data
##############################################################################

library(mice)
data(nhanes, package="mice")

# nhanes data in one chain
imp1 <- mice.1chain( nhanes , burnin=5 , iter=40 , Nimp=4 ,
          imputationMethod=rep("norm" , 4 ) )

# extract first imputed dataset
dati1 <- complete.mids.1chain( imp1 , action=1 )

## End(Not run)
```

`crlrem` R *Utilities: Removing CF Line Endings*

Description

This function removes CF line endings from a text file and writes the processed file in the working directory.

Usage

```
crlrem( filename1 , filename2 )
```

Arguments

`filename1`	Name of the original file (possibly with CF line endings)
`filename2`	Name of the processed file (without CF line endings)

Author(s)

This is code by Dirk Eddelbuettel copied from https://stat.ethz.ch/pipermail/r-devel/2010-September/058480.html

Examples

```
## Not run:
filename1 <- "rm.arraymult__0.02.cpp"
filename2 <- "rm.arraymult__0.03.cpp"
crlrem( filename1 , filename2 )
## End(Not run)
```

---

cxxfunction.copy R *Utilities: Copy of an* Rcpp *File*

Description

Copy the Rcpp function into the working directory.

Usage

```
cxxfunction.copy(cppfct , name)
```

Arguments

cppfct	The Rcpp function
name	Name of the output Rcpp function to be generated

Author(s)

Alexander Robitzsch

References

Eddelbuettel, D. & Francois, R. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, **40(8)**, 1-18.

See Also

See also the cxxfunction in the **inline** package

Examples

```
## Not run:
# define Rcpp file
code1 <- "
    // array A
    Rcpp::NumericMatrix AA(A);
    // Rcpp::IntegerVector dimAA(dimA);
    int nrows = AA.nrow();
    int ncolumns = AA.ncol();
    NumericMatrix Alogis(nrows,ncolumns)  ;
    // compute logistic distribution
```

```
    for (int ii=0; ii<nrows; ++ii){
        NumericVector h1=AA.row(ii) ;
        NumericVector res = plogis( h1 ) ;
        for (int jj=0;jj<ncolumns;++jj){
            Alogis(ii,jj) = res[jj] ;
                        }
                    }

    return( wrap(Alogis) );
    "
# compile Rcpp code
calc1 <- cxxfunction( signature( A= "matrix"), code1, plugin = "Rcpp", verbose=TRUE )
cxxfunction.copy( cppfct=calc1, name="calclogis" )

## End(Not run)
```

`data.allison` *Datasets from Allison's* Missing Data *Book*

Description

Datasets from Allison's missing data book (Allison 2002).

Usage

```
data(data.allison.gssexp)
data(data.allison.hip)
data(data.allison.usnews)
```

Format

- Data `data.allison.gssexp`:

```
'data.frame':   2991 obs. of  14 variables:
$ AGE     : num  33 59 NA 59 21 22 40 25 41 45 ...
$ EDUC    : num  12 12 12 8 13 15 9 12 12 12 ...
$ FEMALE  : num  1 0 1 0 1 1 1 0 1 1 ...
$ SPANKING: num  1 1 2 2 NA 1 3 1 1 NA ...
$ INCOM   : num  11.2 NA 16.2 18.8 13.8 ...
$ NOCHILD : num  0 0 0 0 1 1 0 0 0 0 ...
$ NODOUBT : num  NA NA NA 1 NA NA 1 NA NA 1 ...
$ NEVMAR  : num  0 0 0 0 1 1 0 1 0 0 ...
$ DIVSEP  : num  1 0 0 0 0 0 0 0 0 1 ...
$ WIDOW   : num  0 0 0 0 0 0 1 0 1 0 ...
$ BLACK   : num  1 1 1 0 1 1 0 1 1 1 ...
$ EAST    : num  1 1 1 1 1 1 1 1 1 1 ...
$ MIDWEST : num  0 0 0 0 0 0 0 0 0 0 ...
$ SOUTH   : num  0 0 0 0 0 0 0 0 0 0 ...
```

- Data `data.allison.hip`:

```
'data.frame':   880 obs. of  7 variables:
$ SID : num  1 1 1 1 2 2 2 2 9 9 ...
$ WAVE: num  1 2 3 4 1 2 3 4 1 2 ...
$ ADL : num  3 2 3 3 3 1 2 1 3 3 ...
$ PAIN: num  0 5 0 0 0 1 5 NA 0 NA ...
$ SRH : num  2 4 2 2 4 1 1 2 2 3 ...
$ WALK: num  1 0 0 0 0 0 0 0 1 NA ...
$ CESD: num  9 28 31 11.6 NA ...
```

- Data `data.allison.usnews`:

```
'data.frame':   1302 obs. of  7 variables:
$ CSAT   : num  972 961 NA 881 NA ...
$ ACT    : num  20 22 NA 20 17 20 21 NA 24 26 ...
$ STUFAC : num  11.9 10 9.5 13.7 14.3 32.8 18.9 18.7 16.7 14 ...
$ GRADRAT: num  15 NA 39 NA 40 55 51 15 69 72 ...
$ RMBRD  : num  4.12 3.59 4.76 5.12 2.55 ...
$ PRIVATE: num  1 0 0 0 0 1 0 0 0 1 ...
$ LENROLL: num  4.01 6.83 4.49 7.06 6.89 ...
```

Example Index

Dataset `data.allison.hip`

`data.allison` (Example 1)

Source

The datasets are available from http://www.ats.ucla.edu/stat/examples/md/.

References

Allison, P. D. (2002). *Missing data*. Newbury Park, CA: Sage.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Hip datasets | Imputation using a wide format
#############################################################################

# at first, the hip dataset is 'melted' for imputation

data(data.allison.hip)
  ##   head(data.allison.hip)
  ##     SID WAVE ADL PAIN SRH WALK   CESD
  ##   1   1    1   3    0   2    1  9.000
  ##   2   1    2   2    5   4    0 28.000
  ##   3   1    3   3    0   2    0 31.000
  ##   4   1    4   3    0   2    0 11.579
```

```
  ##   5   2    1   3    0   4    0     NA
  ##   6   2    2   1    1   1    0  2.222

library(reshape)
hip.wide <- reshape::reshape(data.allison.hip, idvar = "SID", timevar = "WAVE",
               direction = "wide")
  ##   > head(hip.wide , 2)
  ##     SID ADL.1 PAIN.1 SRH.1 WALK.1 CESD.1 ADL.2 PAIN.2 SRH.2 WALK.2 CESD.2 ADL.3
  ##   1   1     3      0     2      1      9     2      5     4      0 28.000     3
  ##   5   2     3      0     4      0     NA     1      1     1      0  2.222     2
  ##     PAIN.3 SRH.3 WALK.3 CESD.3 ADL.4 PAIN.4 SRH.4 WALK.4 CESD.4
  ##   1      0     2      0     31     3      0     2      0 11.579
  ##   5      5     1      0     12     1     NA     2      0     NA

# imputation of the hip wide dataset
imp <- mice::mice( as.matrix( hip.wide[,-1] ) , m=5 , maxit=3 )
summary(imp)

## End(Not run)
```

`data.enders` *Datasets from Enders'* Missing Data *Book*

Description

Datasets from Enders' missing data book (2010).

Usage

```
data(data.enders.depression)
data(data.enders.eatingattitudes)
data(data.enders.employee)
```

Format

- Dataset `data.enders.depression`:

```
'data.frame':   280 obs. of  8 variables:
$ txgroup: int  0 0 0 0 0 0 0 0 0 0 ...
$ dep1   : int  46 49 40 47 33 44 45 53 40 55 ...
$ dep2   : int  44 42 28 47 33 41 43 35 43 45 ...
$ dep3   : int  26 29 31 NA 34 34 34 35 35 36 ...
$ r2     : int  0 0 0 0 0 0 0 0 0 0 ...
$ r3     : int  0 0 0 1 0 0 0 0 0 0 ...
$ pattern: int  3 3 3 2 3 3 3 3 3 3 ...
$ dropout: int  0 0 0 1 0 0 0 0 0 0 ...
```

- Dataset `data.enders.eatingattitudes`:

```
'data.frame':   400 obs. of  14 variables:
$ id   : num  1 2 3 4 5 6 7 8 9 10 ...
$ eat1 : num  4 6 3 3 3 4 5 4 4 6 ...
$ eat2 : num  4 5 3 3 2 5 4 3 7 5 ...
$ eat10: num  4 6 2 4 3 4 4 4 6 5 ...
$ eat11: num  4 6 2 3 3 5 4 4 5 5 ...
$ eat12: num  4 6 3 4 3 4 4 4 4 6 ...
$ eat14: num  4 7 2 4 3 4 4 4 6 6 ...
$ eat24: num  3 6 3 3 3 4 4 4 4 5 ...
$ eat3 : num  4 5 3 3 4 4 3 6 4 5 ...
$ eat18: num  5 6 3 5 4 5 3 6 4 6 ...
$ eat21: num  4 5 2 4 4 4 3 5 4 5 ...
$ bmi  : num  18.9 26 18.3 18.2 24.4 ...
$ wsb  : num  9 13 6 5 10 7 11 8 10 12 ...
$ anx  : num  11 19 8 14 7 11 12 12 14 12 ..
```

- Dataset `data.enders.employee`:

```
'data.frame':   480 obs. of  9 variables:
$ id      : num  1 2 3 4 5 6 7 8 9 10 ...
$ age     : num  40 53 46 37 44 39 33 43 35 37 ...
$ tenure  : num  10 14 10 8 9 10 7 9 9 10 ...
$ female  : num  1 1 1 1 1 1 1 1 1 1 ...
$ wbeing  : num  8 6 NA 7 NA 7 NA 7 7 5 ...
$ jobsat  : num  8 5 7 NA 5 NA 5 NA 7 6 ...
$ jobperf : num  6 5 7 5 5 7 7 7 7 6 ...
$ turnover: num  0 0 0 0 0 0 0 0 1 0 ...
$ iq      : num  106 93 107 94 107 118 103 106 108 97 ...
```

Example Index

Dataset `data.enders.employee`

–

Source

The datasets were downloaded from http://www.appliedmissingdata.com/book-examples.html.

References

Enders, C. K. (2010). *Applied missing data analysis*. Guilford Press.

data.graham *Datasets from Grahams* Missing Data *Book*

Description

Datasets from Grahams missing data book (2012).

Usage

```
data(data.graham.ex3)
data(data.graham.ex6)
data(data.graham.ex8a)
data(data.graham.ex8b)
data(data.graham.ex8c)
```

Format

- Dataset data.graham.ex3:

```
'data.frame':   2756 obs. of  20 variables:
$ school  : int  1 1 1 1 1 1 1 1 1 1 ...
$ alc7    : int  1 1 1 7 3 6 1 5 4 3 ...
$ rskreb71: int  1 3 1 2 1 NA 1 2 1 2 ...
$ rskreb72: int  NA NA NA NA NA NA NA 3 2 3 ...
$ rskreb73: int  NA NA NA NA NA NA NA 2 1 2 ...
$ rskreb74: int  NA NA NA NA NA NA NA 3 2 4 ...
$ likepa71: int  4 2 3 3 2 NA 1 4 3 3 ...
$ likepa72: int  5 2 4 2 2 NA 5 3 3 2 ...
$ likepa73: int  4 1 3 3 2 NA 1 3 2 3 ...
$ likepa74: int  5 3 1 5 4 4 3 4 3 2 ...
$ likepa75: int  4 4 4 4 3 3 4 4 3 3 ...
$ posatt71: int  1 1 1 1 1 2 1 NA NA NA ...
$ posatt72: int  1 2 1 1 1 2 4 NA NA NA ...
$ posatt73: int  1 1 1 1 1 2 1 NA NA NA ...
$ alc8    : int  1 8 4 8 5 7 1 3 5 3 ...
$ rskreb81: int  1 4 1 2 2 3 2 3 1 4 ...
$ rskreb82: int  NA NA NA NA NA NA NA 3 1 4 ...
$ rskreb83: int  NA NA NA NA NA NA NA 2 1 2 ...
$ rskreb84: int  NA NA NA NA NA NA NA 3 2 4 ...
$ alc9    : int  3 NA 7 NA 5 7 NA 6 6 7 ...
```

- Dataset data.graham.ex6:

```
'data.frame':   2756 obs. of  9 variables:
$ school  : int  1 1 1 1 1 1 1 1 1 1 ...
$ program : int  0 0 0 0 0 0 0 0 0 0 ...
$ alc7    : int  1 1 1 7 3 6 1 5 4 3 ...
$ riskreb7: int  1 3 1 2 1 NA 1 2 1 2 ...
```

```
$ likepar7: int  4 2 3 3 2 NA 1 4 3 3 ...
$ posatt7 : int  1 1 1 1 1 2 1 NA NA NA ...
$ alc8    : int  1 8 4 8 5 7 1 3 5 3 ...
$ riskreb8: int  1 4 1 2 2 3 2 3 1 4 ...
$ alc9    : int  3 NA 7 NA 5 7 NA 6 6 7 ...
```

- Dataset data.graham.ex8a:

```
'data.frame':   1023 obs. of  20 variables:
$ skill1 : int  28 29 27 29 29 NA NA NA 29 NA ...
$ skill2 : int  NA NA 29 29 NA NA NA NA NA 21 ...
$ skill3 : int  NA NA 29 29 29 NA 28 10 29 25 ...
$ skill4 : int  NA 29 25 29 29 28 29 NA NA NA ...
$ skill5 : int  29 29 28 28 29 NA 29 10 NA 25 ...
$ iplanV1: int  14 18 15 17 16 NA NA NA 18 NA ...
$ iplanV2: int  NA NA 17 16 NA NA NA NA NA 16 ...
$ iplanV3: int  NA NA 16 18 18 NA 17 1 18 16 ...
$ iplanV4: int  NA 18 14 18 14 6 18 NA NA NA ...
$ iplanV5: int  13 18 12 18 18 NA 18 3 NA 5 ...
$ planA1 : int  1 0 2 8 3 NA NA NA 7 NA ...
$ planA2 : int  NA NA 0 4 NA NA NA NA NA 6 ...
$ planA3 : int  NA NA 1 4 7 NA 2 0 1 7 ...
$ planA4 : int  NA 8 0 4 6 0 0 NA NA NA ...
$ planA5 : int  0 7 1 5 7 NA 2 0 NA 6 ...
$ planV1 : int  NA NA NA NA NA NA NA NA NA NA ...
$ planV2 : int  NA NA NA NA NA NA NA NA NA 1 ...
$ planV3 : int  NA NA 1 NA NA NA NA 0 NA 1 ...
$ planV4 : int  NA NA NA NA 2 NA NA NA NA NA ...
$ planV5 : int  2 NA 2 NA NA NA NA 0 NA NA ...
```

- Dataset data.graham.ex8b:

```
'data.frame':   2570 obs. of  6 variables:
$ rskreb71: int  1 3 1 2 1 NA 1 2 1 2 ...
$ rskreb72: int  NA NA NA NA NA NA NA 3 2 3 ...
$ posatt71: int  1 1 1 1 1 2 1 NA NA NA ...
$ posatt72: int  1 2 1 1 1 2 4 NA NA NA ...
$ posatt73: int  1 1 1 1 1 2 1 NA NA NA ...
$ posatt  : int  3 4 3 3 3 6 6 NA NA NA ...
```

- Dataset data.graham.ex8c:

```
'data.frame':   2756 obs. of  16 variables:
$ s1      : int  1 1 1 1 1 1 1 1 1 1 ...
$ s2      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s3      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s4      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s5      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s6      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s7      : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
$ s8      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s9      : int  0 0 0 0 0 0 0 0 0 0 ...
$ s10     : int  0 0 0 0 0 0 0 0 0 0 ...
$ s11     : int  0 0 0 0 0 0 0 0 0 0 ...
$ xalc7   : int  1 1 1 7 3 6 1 5 4 3 ...
$ rskreb72: int  NA NA NA NA NA NA NA 3 2 3 ...
$ likepa71: int  4 2 3 3 2 NA 1 4 3 3 ...
$ posatt71: int  1 1 1 1 1 2 1 NA NA NA ...
$ alc8    : int  1 8 4 8 5 7 1 3 5 3 ...
```

Source

The datasets were downloaded from http://methodology.psu.edu/pubs/books/missing.

References

Graham, J. W. (2012). *Missing data*. New York: Springer.

Examples

```
## Not run:
library(mitools)
library(mice)
library(Amelia)
library(jomo)

#############################################################################
# EXAMPLE 1: data.graham.8a | Imputation under multivariate normal model
#############################################################################

data(data.graham.ex8a)
dat <- data.graham.ex8a
dat <- dat[,1:10]
vars <- colnames(dat)
V <- length(vars)
# remove persons with completely missing data
dat <- dat[ rowMeans( is.na(dat) ) < 1 , ]
summary(dat)

# some descriptive statistics
psych::describe(dat)

#**************
# imputation under a multivariate normal model
M <- 7  # number of imputations

#--------- mice package
# define imputation method
impM <- rep("norm" , V)
names(impM) <- vars
# mice imputation
```

```
imp1a <- mice::mice( dat , imputationMethod=impM , m=M , maxit=4 )
summary(imp1a)
# convert into a list of datasets
datlist1a <- miceadds::mids2datlist(imp1a)

#--------- Amelia package
imp1b <- Amelia::amelia( dat , m=M )
summary(imp1b)
datlist1b <- imp1b$imputations

#--------- jomo package
imp1c <- jomo::jomo1con(Y = dat , nburn=100, nbetween=10, nimp=M)
str(imp1c)
# convert into a list of datasets
datlist1c <- miceadds::jomo2datlist(imp1c)

##############################################################################
# EXAMPLE 2: data.graham.8b | Imputation with categorical variables
##############################################################################

data(data.graham.ex8b)
dat <- data.graham.ex8b
vars <- colnames(dat)
V <- length(vars)

# descriptive statistics
psych::describe(dat)

#*******************************
# imputation in mice using predictive mean matching
imp1a <- mice( dat , m=5 , maxit=10)
datlist1a <- mitools::imputationList( miceadds::mids2datlist(imp1a) )
print(datlist1a)

#*******************************
# imputation in jomo treating all variables as categorical

# Note that variables must have values from 1 to N
# use categorize function from sirt package here
dat.categ <- sirt::categorize( dat , categorical=colnames(dat) , lowest=1 )
dat0 <- dat.categ$data

# imputation in jomo treating all variables as categorical
Y_numcat <- apply( dat0 , 2 , max , na.rm=TRUE )
imp1b <- jomo::jomo1cat(Y_cat = dat0, Y_numcat = Y_numcat, nburn=100,
                nbetween=10, nimp=5)

# recode original categories
datlist1b <- sirt::decategorize( imp1b , categ_design = dat.categ$categ_design )
# convert into a list of datasets
datlist1b <- miceadds::jomo2datlist(datlist1b)
datlist1b <- mitools::imputationList( datlist1b )
```

```
#*****************************
# compare frequency tables for both imputation packages
fun_prop <- function( variable ){
            t1 <- table(variable)
            t1 / sum(t1)
                }

# variable rskreb71
res1a <-  with( datlist1a , fun_prop(rskreb71) )
res1b <-  with( datlist1b , fun_prop(rskreb71) )
summary( miceadds::NMIcombine(qhat = res1a , NMI = FALSE ) )
summary( miceadds::NMIcombine(qhat = res1b , NMI = FALSE ) )

# variable posatt
res2a <-  with( datlist1a , fun_prop(posatt) )
res2b <-  with( datlist1b , fun_prop(posatt) )
summary( miceadds::NMIcombine(qhat = res2a , NMI = FALSE ) )
summary( miceadds::NMIcombine(qhat = res2b , NMI = FALSE ) )

## End(Not run)
```

`data.internet` *Dataset Internet*

Description

Dataset with items corresponding to internet attitudes.

Usage

```
data(data.internet)
```

Format

A data frame with 281 observations on the following 22 variables.

The format of the dataset is

```
'data.frame':   281 obs. of  22 variables:
$ IN1 : num  1 5 2 3 1 3 2 3 2 1 ...
$ IN2 : num  4 3 2 7 7 4 4 7 4 3 ...
$ IN3 : num  4 5 4 2 1 2 5 2 2 4 ...
[...]
$ IN20: num  3 2 2 3 3 4 2 7 2 2 ...
$ IN21: num  3 3 6 5 4 4 5 5 6 5 ...
$ IN22: num  3 4 2 5 3 5 3 7 3 5 ...
```

Details

The following text is copied from http://people.few.eur.nl/groenen/Data/index.htm

The data set is based on a questionnaire on attitudes towards the Internet. It consists of evaluations of 22 statements about the Internet by 281 students at Erasmus University Rotterdam. These data were gathered around 2002 before the wide availability of broadband Internet access in the Netherlands. The statements were evaluated using a seven-point Likert scale, ranging from 1 (completely disagree) to 7 (completely agree).

We would like to thank Peter Verhoef for making these data available.

Each variable (statement) is coded as follows:

1. Completely disagree
2. Disagree
3. Slightly disagree
4. Neutral
5. Slightly agree
6. Agree
7. Completely agree

Internet items:

1. Paying using Internet is safe
2. Surfing the Internet is easy
3. Internet is unreliable
4. Internet is slow
5. Internet is user-friendly
6. Internet is the future's means of communication
7. Internet is addictive
8. Internet is fast
9. Sending personal data using the Internet is unsafe
10. The prices of Internet subscriptions are high
11. Internet offers many possibilities for abuse
12. The costs of surfing are high
13. Internet offers unbounded opportunities
14. Internet phone costs are high
15. The content of web sites should be regulated
16. Internet is easy to use
17. I like surfing
18. I often speak with friends about the Internet
19. I like to be informed of important new things
20. I always attempt new things on the Internet first
21. I regularly visit websites recommended by others
22. I know much about the Internet

Example Index

ma.scale2 (Example 1), mice.impute.2l.pls2 (Example 1), pca.covridge (Example 1), tw.imputation (Example 1)

Source

Peter Verhoef

http://people.few.eur.nl/groenen/Data/index.htm

Examples

```
data(data.internet)
# missing proportions
colMeans( is.na(data.internet) )
```

`data.largescale` *Large-scale Dataset for Testing Purposes (Many Cases, Few Variables)*

Description

Large-scale dataset with many cases and few variables included for testing purposes.

Usage

```
data(data.largescale)
```

Format

A data frame with 14000 observations on the following 13 variables. The format is

```
'data.frame':   14000 obs. of  13 variables:
$ id: num  1e+07 1e+07 1e+07 1e+07 1e+07 ...
$ D1: num  0 0 0 0 1 0 0 0 0 0 ...
$ D2: num  0 0 0 1 0 1 0 1 0 0 ...
$ D3: num  0 0 0 0 0 0 0 0 0 0 ...
$ D4: num  0 0 0 1 0 0 0 1 0 0 ...
$ D5: num  0 0 0 0 0 1 0 0 0 0 ...
$ v1: num  118 117 94 106 86 117 96 96 82 95 ...
$ v2: num  101 101 86 101 65 94 72 75 70 99 ...
$ v3: num  0 0 0 0 0 1 0 0 0 0 ...
$ v4: num  3 NA 3 5 2 5 5 5 4 2 ...
$ v5: num  0 NA 0 0 0 1 0 0 0 0 ...
$ v6: num  3 3 3 4 NA 1 3 3 2 3 ...
$ v7: num  51 36 14 47 22 17 13 37 47 38 ...
```

data.ma *Example Datasets for* **miceadds** *Package*

Description

Example datasets for **miceadds** package.

Usage

```
data(data.ma01)
data(data.ma02)
data(data.ma03)
data(data.ma04)
data(data.ma05)
data(data.ma06)
```

Format

- Dataset data.ma01:
  Dataset with students nested within school and student weights (studwgt). The format is
  ```
  'data.frame':   4073 obs. of  11 variables:
  $ idstud  : num  1e+07 1e+07 1e+07 1e+07 1e+07 ...
  $ idschool: num  1001 1001 1001 1001 1001 ...
  $ studwgt : num  6.05 6.05 5.27 5.27 6.05 ...
  $ math    : int  594 605 616 524 685 387 536 594 387 562 ...
  $ read    : int  647 651 539 551 689 502 503 597 580 576 ...
  $ migrant : int  0 0 0 1 0 0 1 0 0 0 ...
  $ books   : int  6 6 5 2 6 3 4 6 6 5 ...
  $ hisei   : int  NA 77 69 45 66 53 43 NA 64 50 ...
  $ paredu  : int  3 7 7 2 7 3 4 NA 7 3 ...
  $ female  : int  1 1 0 0 1 1 0 0 1 1 ...
  $ urban   : num  1 1 1 1 1 1 1 1 1 1 ...
  ```

- Dataset data.ma02:
  10 multiply imputed datasets of incomplete data data.ma01. The format is
  ```
  List of 10
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  $ :'data.frame':       4073 obs. of  11 variables:
  ```

- Dataset data.ma03:

  This dataset contains one variable math_EAP for which a conditional posterior distribution with EAP and its associated standard deviation is available.

```
'data.frame':   120 obs. of  8 variables:
$ idstud    : int  1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...
$ female    : int  0 1 1 1 1 0 1 1 1 1 ...
$ migrant   : int  1 1 0 1 1 0 0 0 1 0 ...
$ hisei     : int  44 NA 26 NA 32 60 31 NA 34 26 ...
$ educ      : int  NA 2 NA 1 4 NA 2 NA 2 NA ...
$ read_wle  : num  74.8 78.1 103.2 81.2 119.2 ...
$ math_EAP  : num  337 342 264 285 420 ...
$ math_SEEAP: num  28 29.5 28.6 28.5 27.5 ...
```

- Dataset data.ma04:

  This dataset contains two hypothetical scales A and B and single variables V5, V6 and V7.

```
'data.frame':   281 obs. of  13 variables:
$ group: int  1 1 1 1 1 1 1 1 1 1 ...
$ A1   : int  2 2 2 1 1 3 3 NA 2 1 ...
$ A2   : int  2 2 2 3 1 2 4 4 4 4 ...
$ A3   : int  2 3 3 4 1 3 2 2 2 4 ...
$ A4   : int  3 4 6 4 7 5 3 5 5 1 ...
$ V5   : int  2 2 5 5 4 3 4 1 3 4 ...
$ V6   : int  2 5 5 1 1 3 2 2 2 4 ...
$ V7   : int  6 NA 4 5 6 2 5 5 6 7 ...
$ B1   : int  7 NA 6 4 5 2 5 7 3 7 ...
$ B2   : int  6 NA NA 6 3 3 4 6 6 7 ...
$ B3   : int  7 NA 7 4 3 4 3 7 5 NA ...
$ B4   : int  4 5 6 5 4 3 4 5 2 1 ...
$ B5   : int  7 NA 7 4 4 3 5 7 5 4 ...
```

- Dataset data.ma05:

  This is a two-level dataset with students nested within classes. Variables at the student level are Dscore, Mscore, denote, manote, misei and migrant. Variables at the class level are sprengel and groesse.

```
'data.frame':   1673 obs. of  10 variables:
$ idstud  : int  100110001 100110002 100110003 100110004 100110005  ...
$ idclass : int  1001 1001 1001 1001 1001 1001 1001 1001 1001 1001 ...
$ Dscore  : int  NA 558 643 611 518 552 NA 534 409 543 ...
$ Mscore  : int  404 563 569 621 653 651 510 NA 517 566 ...
$ denote  : int  NA 1 1 1 3 2 3 2 3 2 ...
$ manote  : int  NA 1 1 1 1 1 2 2 2 1 ...
$ misei   : int  NA 51 NA 38 NA 50 53 53 38 NA ...
$ migrant : int  NA 0 0 NA 0 0 0 0 0 NA ...
$ sprengel: int  0 0 0 0 0 0 0 0 0 0 ...
$ groesse : int  25 25 25 25 25 25 25 25 25 25 ...
```

- Dataset data.ma06:

This is a dataset in which the variable FC is only available with grouped values (coarse data or interval data).

```
'data.frame':   198 obs. of  7 variables:
$ id    : num  1001 1002 1003 1004 1005 ...
$ A1    : int  14 7 10 15 0 5 9 6 8 0 ...
$ A2    : int  5 6 4 8 2 5 4 0 7 0 ...
$ Edu   : int  4 3 1 5 5 1 NA 1 5 3 ...
$ FC    : int  3 2 2 2 2 NA NA 2 2 NA ...
$ FC_low: num  10 5 5 5 5 0 0 5 5 0 ...
$ FC_upp: num  15 10 10 10 10 100 100 10 10 100 ...
```

Example Index

Dataset data.ma01

mice.1chain (Example 3), mice.impute.weighted.pmm (Example 1), ma.wtd.statNA (Example 1)

Dataset data.ma02

fast.groupmean (Example 1),

Dataset data.ma03

mice.impute.2l.eap (Example 1)

Dataset data.ma04

mice.impute.2l.plausible.values (Example 1)

Dataset data.ma05

mice.impute.2l.contextual.pmm (Example 1), mice.impute.2l.latentgroupmean (Example 1)

Dataset data.ma06

mice.impute.grouped (Example 1),

---

data.smallscale *Small-Scale Dataset for Testing Purposes (Moderate Number of Cases, Many Variables)*

Description

Small-scale dataset for testing purposes (moderate number of cases, many variables)

Usage

```
data(data.smallscale)
```

Format

A data frame with 675 observations on the following 164 variables. The format is

```
'data.frame':   675 obs. of  164 variables:
$ v1  : num  3 3 2 3 3 0 1 0 3 NA ...
$ v2  : num  3 0 1 3 0 0 0 3 2 NA ...
$ v3  : num  0 0 2 3 2 0 1 0 0 NA ...
$ v4  : num  1 3 3 3 NA 0 0 0 3 NA ...
$ v5  : num  0 0 3 3 0 0 3 1 3 3 ...
$ v6  : num  8 8 9 8 9 9 9 8 9 9 ...
[...]
```

`datalist2mids` *Converting a List of Multiply Imputed Data Sets into a* `mids` *Object*

Description

This function converts a list of multiply imputed data sets to a `mice::mids` object.

Usage

```
datalist2mids(dat.list, progress = TRUE)
```

Arguments

`dat.list`	List of multiply imputed data sets
`progress`	An optional logical indicating whether conversion process be displayed

Value

An object of class `mids`

Author(s)

Alexander Robitzsch

See Also

See `mice::as.mids` for converting a multiply imputed dataset in long format into a `mids` object.

Examples

```
##############################################################################
# EXAMPLE 1: Imputation of NHANES data using Amelia package
##############################################################################

library(mice)
library(Amelia)

data(nhanes,package="mice")
set.seed(566)  # fix random seed

# impute 10 datasets using Amelia
a.out <- Amelia::amelia(x =  nhanes , m=10)
# plot of observed and imputed data
plot(a.out)

# convert list of multiply imputed datasets into a mids object
a.mids <- datalist2mids( a.out$imputations )

# linear regression: apply mice functionality lm.mids
mod <- with( a.mids , lm( bmi ~ age ) )
summary( pool( mod ) )
  ##                    est      se         t       df     Pr(>|t|)    lo 95
  ##   (Intercept) 30.718881 2.22960 13.777753 12.58135 5.830925e-09 25.88578
  ##   age         -2.435746 1.08551 -2.243872 14.93153 4.043506e-02 -4.75038
  ##                    hi 95 nmis       fmi    lambda
  ##   (Intercept) 35.5519834   NA 0.4013689 0.3132139
  ##   age         -0.1211117    0 0.3153636 0.2294162

## Not run:
# fit linear regression model in Zelig
library(Zelig)
mod2 <- Zelig::zelig( bmi ~ age , model="ls" , data=a.out$imputations , cite=FALSE)
summary(mod2)
  ##   > summary(mod2)
  ##
  ##     Model: ls
  ##     Number of multiply imputed data sets: 10
  ##
  ##   Combined results:
  ##
  ##   Call:
  ##   lm(formula = formula, weights = weights, model = F, data = data)
  ##
  ##   Coefficients:
  ##                   Value Std. Error    t-stat      p-value
  ##   (Intercept) 30.718881    2.22960 13.777753 4.603995e-24
  ##   age         -2.435746    1.08551 -2.243872 2.612377e-02
  ##
  ##   For combined results from datasets i to j, use summary(x, subset = i:j).
  ##   For separate results, use print(summary(x), subset = i:j).
```

```
# fit linear regression using mitools package
library(mitools)
datimp <- mitools::imputationList(a.out$imputations)
mod3 <- with( datimp, lm( bmi ~ age ) )
summary( mitools::MIcombine( mod3 ) )
  ##  Multiple imputation results:
  ##        with(datimp, lm(bmi ~ age))
  ##        MIcombine.default(mod3)
  ##                results      se    (lower     upper) missInfo
  ##  (Intercept) 30.718881 2.22960 26.290536 35.1472266     33
  ##  age         -2.435746 1.08551 -4.578471 -0.2930208     24

## End(Not run)
```

`draw.pv.ctt` *Plausible Value Imputation Using a Known Measurement Error Variance (Based on Classical Test Theory)*

Description

This function provides unidimensional plausible value imputation with a known measurement error variance or classical test theory (Mislevy, 1991). The reliability of the scale is estimated by Cronbach's Alpha or can be provided by the user.

Usage

```
draw.pv.ctt(y, dat.scale = NULL, x=NULL, samp.pars = TRUE,
      alpha = NULL, sig.e = NULL, var.e=NULL , true.var = NULL)
```

Arguments

`y`	Vector of scale scores if `y` should not be used.
`dat.scale`	Matrix of item responses
`x`	Matrix of covariates
`samp.pars`	An optional logical indicating whether scale parameters (reliability or measurement error standard deviation) should be sampled
`alpha`	Reliability estimate of the scale. The default of `NULL` means that Cronbach's alpha will be used as a reliability estimate.
`sig.e`	Optional vector of the standard deviation of the error. Note that it is *not* the error variance.
`var.e`	Optional vector of the variance of the error.
`true.var`	True score variance

Details

The linear model is assumed for drawing plausible values of a variable Y contaminated by measurement error. Assuming $Y = \theta + e$ and a linear regression model for θ

$$\theta = \boldsymbol{X}\beta + \epsilon$$

(plausible value) imputations from the posterior distribution $P(\theta|Y, \boldsymbol{X})$ are drawn. See Mislevy (1991) for details.

Value

A vector with plausible values

Note

Plausible value imputation is also labeled as multiple overimputation (Blackwell, Honaker & King, 2011).

Author(s)

Alexander Robitzsch

References

Blackwell, M., Honaker, J., & King, G. (2011). *Multiple overimputation: A unified approach to measurement error and missing data*. Technical Report.

Mislevy, R. J. (1991). Randomization-based inference about latent variables from complex samples. *Psychometrika*, **56**, 177-196.

See Also

See also `sirt::plausible.value.imputation.raschtype` for plausible value imputation.

Examples

```
#############################################################################
# SIMULATED EXAMPLE 1: Scale scores
#############################################################################

set.seed(899)
n <- 5000       # number of students
x <- round( runif( n , 0 ,1 ) )
y <- rnorm(n)
# simulate true score theta
theta <- .6 + .4*x + .5 * y + rnorm(n)
# simulate observed score by adding measurement error
sig.e <- rep( sqrt(.40) , n )
theta_obs <- theta + rnorm( n , sd=sig.e)

# calculate alpha
( alpha <- var( theta ) / var( theta_obs ) )
```

```
# [1] 0.7424108
# => Ordinarily, sig.e or alpha will be known, assumed or estimated by using items,
#    replications or an appropriate measurement model.

# create matrix of predictors
X <- as.matrix( cbind(x , y ) )

# plausible value imputation with scale score
imp1 <- draw.pv.ctt( y=theta_obs , x = X , sig.e =sig.e )
# check results
lm( imp1 ~ x + y )

# imputation with alpha as an input
imp2 <- draw.pv.ctt( y=theta_obs , x = X , alpha = .74 )
lm( imp2 ~ x + y )
```

`fast.groupmean` *Calculation of Groupwise Descriptive Statistics for Matrices*

Description

Calculates some groupwise descriptive statistics

Usage

```
fast.groupmean(data, group, weights=NULL, extend=FALSE)

fast.groupsum(data, group, weights=NULL, extend=FALSE)
```

Arguments

`data`	A numeric data frame
`group`	A vector of group identifiers
`weights`	An optional vector of sample weights
`extend`	Optional logical indicating whether the group means (or sums) should be extended to the original dimensions of the dataset.

Value

A data frame with groupwise calculated statistics

Author(s)

Alexander Robitzsch

Examples

```
#############################################################################
# EXAMPLE 1: Group means data.ma02
#############################################################################

data( data.ma02 )
dat <- data.ma02[[1]] # select first dataset

# group means for read and math
fast.groupmean( dat[ , c("read","math") ] , group=dat$idschool )


# extend group means to original dataset
fast.groupmean( dat[ , c("read","math") ] , group=dat$idschool , extend =TRUE )
```

`grep.vec` R *Utilities: Vector Based Version of* `grep`

Description

This function imitates the usage of `grep` but it is extended to a vector argument.

Usage

```
grep.vec(pattern.vec, x, operator="AND")
```

Arguments

`pattern.vec`	String which should be looked for in vector `x`
`x`	A character vector
`operator`	An optional string. The default argument `"AND"` searches all entries in `x` which contain all elements of `pattern.vec`. If `operator` is different from the default, then the `"OR"` logic applies, i.e. the functions searches for vector entries which contain at least one of the strings in `pattern.vec`.

Author(s)

Alexander Robitzsch

Examples

```
vec <- c("abcd" , "bcde" , "aedf" , "cdf" )
# search for entries in vec with contain 'a' and 'f'
#  -> operator = "AND"
grep.vec( pattern.vec=c("a","f") , x=vec )
  ##   $x
  ##   [1] "aedf"
  ##   $index.x
```

```
  ##   [1] 3

# search for entries in vec which contain 'a' or 'f'
grep.vec( pattern.vec=c("a","f") , x=vec , operator="OR")
  ##   $x
  ##   [1] "abcd" "aedf" "cdf"
  ##   $index.x
  ##   [1] 1 3 4
```

---

index.dataframe R *Utilities: Include an Index to a Data Frame*

Description

This function includes an index variable to a data frame in the first column.

Usage

```
index.dataframe(data,systime=FALSE)
```

Arguments

data	Data frame
systime	Should system time be included in the second column of the data frame?

Author(s)

Alexander Robitzsch

Examples

```
dfr <- matrix( 2*1:12-3 , 4 ,3 )
colnames(dfr) <- paste0("X",1:ncol(dfr))
index.dataframe( dfr)
  ##     index X1 X2 X3
  ##   1     1 -1  7 15
  ##   2     2  1  9 17
  ##   3     3  3 11 19
  ##   4     4  5 13 21
index.dataframe( dfr , systime=TRUE)
  ##     index         file_created X1 X2 X3
  ##   1     1  2013-08-22 10:26:28 -1  7 15
  ##   2     2  2013-08-22 10:26:28  1  9 17
  ##   3     3  2013-08-22 10:26:28  3 11 19
  ##   4     4  2013-08-22 10:26:28  5 13 21
```

jomo2datlist *Converts a* **jomo** *Data Frame in Long Format into a List of Datasets*

Description

Converts a **jomo** data frame in long format into a list of datasets.

Usage

```
jomo2datlist(jomo.dataframe, variable = "Imputation")
```

Arguments

jomo.dataframe	Data frame generated in **jomo** package
variable	Variable name for imputation index

Value

List of multiply imputed datasets

Author(s)

Alexander Robitzsch

See Also

See the **jomo** package.

Examples

```
#############################################################################
# EXAMPLE 1: Dataset nhanes | jomo imputation and conversion into a data list
#############################################################################

data( nhanes, package="mice")
dat <- nhanes

# impute under multivariate normal model in jomo
imp1 <- jomo::jomo1con(Y = dat , nburn=100, nbetween=10, nimp=5)
# convert into a list of datasets
datlist1 <- jomo2datlist(imp1)
```

`kernelpls.fit2` *Kernel PLS Regression*

Description

Fits a PLS regression model with the kernel algorithm (Dayal & Macgregor, 1997).

Usage

```
kernelpls.fit2(X, Y, ncomp)

## S3 method for class 'kernelpls.fit2'
predict(object,X, ...)
```

Arguments

`X`	Matrix of regressors
`Y`	Vector of a univariate outcome
`ncomp`	Number of components to be extracted
`object`	Object of class `kernelpls.fit2`
`...`	Further arguments to be passed

Value

The same list as in `pls::kernelpls.fit` of the **pls** package is produced.

In addition, R^2 measures are contained in `R2`.

Author(s)

Alexander Robitzsch

This code is a **Rcpp** translation of the original `pls::kernelpls.fit` function from the **pls** package (see Mevik & Wehrens, 2007).

References

Dayal, B., & Macgregor, J. F. (1997). Improved PLS algorithms. *Journal of Chemometrics*, **11**, 73-85.

Mevik, B. H., & Wehrens, R. (2007). The **pls** package: Principal component and partial least squares regression in R. *Journal of Statistical Software*, **18**, 1-24.

See Also

See the **pls** package for further estimation algorithms.

Examples

```
#############################################################################
# SIMULATED EXAMPLE 1: 300 cases on 100 variables
#############################################################################
set.seed(789)

N <- 300        # number of cases
p <- 100        # number of predictors
rho1 <- .6      # correlations between predictors

# simulate data
Sigma <- diag(1-rho1,p) + rho1
X <- mvtnorm::rmvnorm( N , sigma=Sigma )
beta <- seq( 0 , 1 , len=p )
y <- ( X %*% beta )[,1] + rnorm( N , sd = .6 )
Y <- matrix(y,nrow=N , ncol=1 )

# PLS regression
res <- kernelpls.fit2( X=X , Y = Y , ncomp=20 )

# predict new scores
Xpred <- predict( res , X = X[1:10,] )

## Not run:
#############################################################################
# EXAMPLE 2: Dataset yarn from pls package
#############################################################################

# use kernelpls.fit from pls package
library(pls)
data(yarn,package="pls")
mod1 <- pls::kernelpls.fit( X = yarn$NIR , Y = yarn$density , ncomp = 10 )
# use kernelpls.fit2 from miceadds package
Y <- matrix( yarn$density, ncol=1 )
mod2 <- kernelpls.fit2( X = yarn$NIR , Y = Y , ncomp = 10 )

## End(Not run)
```

`library_install` R *Utilities: Loading a Package or Installation of a Package if Necessary*

Description

Loads packages specified in vector pkg. If some packages are not yet installed, they will be automatically installed by this function using `install.packages`.

Usage

```
library_install( pkg , ... )
```

Arguments

pkg	Vector with package names
...	Further arguments to be passed to `install.packages`

Author(s)

Alexander Robitzsch

Examples

```
## Not run:
# try to load packages PP and MCMCglmm
library_install( pkg =  c("PP" , "MCMCglmm") )

## End(Not run)
```

---

lm.cluster *Cluster Robust Standard Errors for Linear Models and General Linear Models*

Description

Computes cluster robust standard errors for linear model (`stats::lm`) and general linear models (`stats::glm`) using the `multiwayvcov::cluster.vcov` function in the **multiwayvcov** package.

Usage

```
lm.cluster(data, formula, cluster, ...)

glm.cluster(data, formula, cluster, ...)

## S3 method for class 'lm.cluster'
summary(object,...)
## S3 method for class 'glm.cluster'
summary(object,...)

## S3 method for class 'lm.cluster'
coef(object,...)
## S3 method for class 'glm.cluster'
coef(object,...)

## S3 method for class 'lm.cluster'
vcov(object,...)
## S3 method for class 'glm.cluster'
vcov(object,...)
```

Arguments

data	Data frame
formula	An R formula
cluster	Variable name for cluster variable contained in data or a vector with cluster identifiers
...	Further arguments to be passed to stats::lm and stats::glm
object	Object of class lm.cluster or glm.cluster

Value

List with following entries

lm_res	Value of stats::lm
glm_res	Value of stats::glm
vcov	Covariance matrix of parameter estimates

Author(s)

Alexander Robitzsch

See Also

stats::lm, stats::glm, multiwayvcov::cluster.vcov

Examples

```
#############################################################################
# EXAMPLE 1: Cluster robust standard errors data.ma01
#############################################################################

data(data.ma01)
dat <- data.ma01

#*** Model 1: Linear regression
mod1 <- lm.cluster( data = dat , formula = read ~ hisei + female ,
               cluster = "idschool" )
coef(mod1)
vcov(mod1)
summary(mod1)

# estimate Model 1, but cluster is provided as a vector
mod1b <- lm.cluster( data = dat, formula = read ~ hisei + female,
                 cluster = dat$idschool)
summary(mod1b)

#*** Model 2: Logistic regression
dat$highmath <- 1 * ( dat$math > 600 )   # create dummy variable
mod2 <- glm.cluster( data = dat , formula = highmath ~ hisei + female ,
                cluster = "idschool" , family="binomial")
```

```
coef(mod2)
vcov(mod2)
summary(mod2)

## Not run:
#############################################################################
# EXAMPLE 2: Cluster robust standard errors for multiply imputed datasets
#############################################################################

library(mitools)
data(data.ma05)
dat <- data.ma05

# imputation of the dataset: use just a single imputation for simplicity
resp <- dat[ , - c(1:2) ]
imp <- mice::mice( resp , imputationMethod="norm" , maxit=3 , m=6 )
datlist <- mids2datlist( imp )

# linear regression with cluster robust standard errors
mod <- lapply(  datlist, FUN = function(data){
          lm.cluster( data=data , formula=denote ~ migrant+ misei , cluster = dat$idclass  )
                                  }  )
# extract parameters and covariance matrix
betas <- lapply( mod , FUN = function(rr){ coef(rr) } )
vars <- lapply( mod , FUN = function(rr){ vcov(rr) } )
# conduct statistical inference
summary( mitools::MIcombine(betas,vars) )

## End(Not run)
```

---

load.data R *Utilities: Loading/Reading Data Files using* **miceadds**

Description

This function is a wrapper function for loading or reading data frames or matrices.

Usage

```
load.data( filename , type="Rdata" , path=getwd() , spss.default=TRUE , ...)
```

Arguments

filename	Name of the data file (matrix or data frame). This can also be a part of the file name and the most recent file is loaded.
type	The type of file in which the data frame or matrix should be loaded. This can be Rdata (for R binary format, using load.Rdata2), csv (using utils::read.csv2), csv2 (using utils::read.csv), table (using utils::read.table; the dataset must have the file extension dat or txt) or sav (using foreign::read.spss).

path	Directory from which the dataset should be loaded
spss.default	Optional logical which is only applied for type="sav" indicating whether the arguments to.data.frame=TRUE and use.value.labels=FALSE are used.
...	Further arguments to be passed to load.Rdata2, read.csv2, read.csv, read.table or foreign::read.spss.

Author(s)

Alexander Robitzsch

See Also

See also load.Rdata for loading R data frames.

See save.Rdata and save.data for saving/writing R data frames.

Examples

```
## Not run:
# load a data frame in the file "data_s3.Rdata" and save this
# as the object "dat.s3"
dat.s3 <- load.data( filename = "data_s3.Rdata" , type = "Rdata" )

## End(Not run)
```

---

load.Rdata R *Utilities: Loading* Rdata *Files in a Convenient Way*

Description

These functions loads a Rdata object saved as a data frame or a matrix in the current R environment. The function load.Rdata saves the loaded object in the global environment while load.Rdata2 loads the object only specified environments. Hence, usage of load.Rdata2 instead of load.Rdata is recommended.

Usage

```
load.Rdata(filename, objname)

load.Rdata2(filename, path=getwd())
```

Arguments

filename	Rdata file (matrix or data frame)
objname	Object name. This object will be a global variable in R.
path	Directory from which the dataset should be loaded

Author(s)

Alexander Robitzsch

See Also

See also save.Rdata for saving data frames in a Rdata format.

See also: base::load, base::save

Examples

```
## Not run:
# load a data frame in the file "data_s3.Rdata" and save this
# as the object "dat.s3"
load.Rdata( filename = "data_s3.Rdata" , "dat.s3" )
head(dat.s3)

# Alternatively one can use the function
dat.s3 <- load.Rdata2( filename = "data_s3.Rdata")

## End(Not run)
```

---

ma.scale2 *Standardization of a Matrix*

Description

This function performs a z-standardization for a numeric matrix. Note that in a case of a zero standard deviation all matrix entries are divided by a small number such that no NaNs occur.

Usage

```
ma.scale2(x, missings=FALSE)
```

Arguments

x	A numeric matrix in which missing values are permitted
missings	A logical indicating whether missings occur (or could occur) in the dataset

Value

A matrix

Author(s)

Alexander Robitzsch

See Also

scale

Examples

```
#############################################################################
# EXAMPLE 1: z-standardization data.internet
#############################################################################

data(data.internet)
dat <- data.internet

# z-standardize all variables in this dataset
zdat <- ma.scale2( dat , missings=TRUE )

## Not run:
#############################################################################
# SIMULATED EXAMPLE 2: Speed comparison for many cases and many variables
#############################################################################

set.seed(9786)
# 3000 cases, 200 variables
N <- 3000 ; p <- 200
# simulate some data
x <- matrix( rnorm( N*p ) , N , p )
x <- round( x , 2 )

# compare computation times for 10 replications
B <- 10
    s1 <- Sys.time() # scale in R
for (bb in 1:B){
    res <- scale(x)
} ; s2 <- Sys.time() ; d1 <- s2-s1

    s1 <- Sys.time() # scale in miceadds
for (bb in 1:B){
    res1 <- ma.scale2(x)
} ; s2 <- Sys.time() ; d2 <- s2-s1

# scale in miceadds with missing handling
    s1 <- Sys.time()
for (bb in 1:B){
    res1 <- ma.scale2(x,missings=TRUE)
} ; s2 <- Sys.time() ; d3 <- s2-s1
d1      # scale in R
d2      # scale in miceadds (no missing handling)
d3      # scale in miceadds (with missing handling)
  ##   > d1      # scale in R
  ##   Time difference of 1.622431 secs
  ##   > d2      # scale in miceadds (no missing handling)
  ##   Time difference of 0.156003 secs
  ##   > d3      # scale in miceadds (with missing handling)
  ##   Time difference of 0.2028039 secs

## End(Not run)
```

`ma.wtd.statNA` *Some Multivariate Descriptive Statistics for Weighted Data in* **miceadds**

Description

Some multivariate descriptive statistics for weighted data in **miceadds**.

Usage

```
ma.wtd.meanNA(data, weights = rep(1, nrow(data)) )
ma.wtd.sdNA(data, weights = rep(1, nrow(data)) )
ma.wtd.covNA(data, weights = rep(1, nrow(data)) )
ma.wtd.corNA(data, weights = rep(1, nrow(data)) )
```

Arguments

`data`	Numeric data frame
`weights`	Optional vector of sampling weights

Details

Contrary to ordinary R practice, missing values are ignored in the calculation of descriptive statistics.

`ma.wtd.meanNA`	weighted means
`ma.wtd.sdNA`	weighted standard deviations
`ma.wtd.covNA`	weighted covariance matrix
`ma.wtd.corNA`	weighted correlation matrix

Value

A matrix or a vector depending on the requested statistic.

Author(s)

Alexander Robitzsch

Examples

```
#############################################################################
# EXAMPLE 1: Weighted statistics data.ma01
#############################################################################

data(data.ma01)
dat <- as.matrix(data.ma01[,-c(1:3)])

# weighted mean
```

```
ma.wtd.meanNA( dat , weights=data.ma01$studwgt  )

# weighted SD
ma.wtd.sdNA( dat , weights=data.ma01$studwgt  )

# weighted covariance
ma.wtd.covNA( dat , weights=data.ma01$studwgt )

# weighted correlation
ma.wtd.corNA( dat , weights=data.ma01$studwgt )
```

`mi.anova` *Analysis of Variance for Multiply Imputed Data Sets (Using the D_2 Statistic)*

Description

This function combines F values from analysis of variance using the D_2 statistic which is based on combining χ^2 statistics (see Allison, 2001; `micombine.F`, `micombine.chisquare`).

Usage

```
mi.anova(mi.res, formula, type=2)
```

Arguments

`mi.res`	Object of class `mids` or `mids.1chain`
`formula`	Formula for `lm` function. Note that this can be also a string.
`type`	Type for ANOVA calculations. For `type=3`, the `Anova` function form the **car** package is used.

Value

A list with the following entries:

`r.squared`	Explained variance R^2
`anova.table`	ANOVA table

Author(s)

Alexander Robitzsch

References

Allison, P. D. (2002). *Missing data*. Newbury Park, CA: Sage.

See Also

`micombine.F`, `micombine.chisquare`

Examples

```
#############################################################################
# EXAMPLE 1: nhanes2 data | two-way ANOVA
#############################################################################

library(mice)
library(car)
data(nhanes2, package="mice")
set.seed(9090)

# nhanes data in one chain and 8 imputed datasets
mi.res <- mice.1chain( nhanes2 , burnin=4 , iter=20 , Nimp=8 )
# 2-way analysis of variance (type 2)
an2a <- mi.anova(mi.res=mi.res, formula="bmi ~ age * chl" )
# 2-way analysis of variance (type 3)
an2b <- mi.anova(mi.res=mi.res, formula="bmi ~ age * chl" , type=3)

#****** analysis based on first imputed dataset

# extract first dataset
dat1 <- complete( mi.res$mids )

# type 2 ANOVA
lm1 <- lm( bmi ~ age * chl , data = dat1 )
summary( aov( lm1 ) )
# type 3 ANOVA
lm2 <- lm( bmi ~ age * chl , data= dat1,    contrasts=list(age=contr.sum))
car::Anova( lm2 , type=3)
```

`mice.1chain` *Multiple Imputation by Chained Equations using One Chain*

Description

This function modifies the `mice::mice` function to multiply impute a dataset using a long chain instead of multiple parallel chains which is the approach employed in `mice::mice`.

Usage

```
mice.1chain(data, burnin = 10, iter = 20, Nimp = 10,
   method = vector("character", length = ncol(data)),
   predictorMatrix = (1 - diag(1, ncol(data))),
   visitSequence = (1:ncol(data))[apply(is.na(data), 2, any)],
   form = vector("character", length = ncol(data)),
   post = vector("character", length = ncol(data)),
   defaultMethod = c("pmm", "logreg", "polyreg", "polr"),
   diagnostics = TRUE, printFlag = TRUE, seed = NA, imputationMethod = NULL,
   defaultImputationMethod = NULL, data.init = NULL, ...)
```

```
## S3 method for class 'mids.1chain'
summary(object,...)

## S3 method for class 'mids.1chain'
plot(x,plot.burnin=FALSE , ask=TRUE , ...)
```

Arguments

data	Numeric matrix
burnin	Number of burn-in iterations
iter	Total number of imputations (larger than burnin)
Nimp	Number of imputations
method	See mice::mice
predictorMatrix	See mice::mice
visitSequence	See mice::mice
form	See mice::mice
post	See mice::mice
defaultMethod	See mice::mice
diagnostics	See mice::mice
printFlag	See mice::mice
seed	See mice::mice
imputationMethod	See mice::mice
defaultImputationMethod	See mice::mice
data.init	See mice::mice
object	Object of class mids.1chain
x	Object of class mids.1chain
plot.burnin	An optional logical indicating whether burnin iterations should be included in the traceplot
ask	An optional logical indicating a user request for viewing next plot
...	See mice::mice

Value

A list with following entries

midsobj	Objects of class mids
datlist	List of multiply imputed datasets
datalong	Original and imputed dataset in the long format
implist	List of mids objects for every imputation
chainMpar	Trace of means for all imputed variables
chainVarpar	Trace of variances for all imputed variables

Note

Multiple imputation can also be used for determining causal effects (see Example 3; Schafer & Kang, 2008).

Author(s)

Alexander Robitzsch

See Also

mice::mice

Examples

```
#############################################################################
# EXAMPLE 1: One chain nhanes data
#############################################################################

library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes data in one chain
imp.mi1 <- mice.1chain( nhanes , burnin=5 , iter=40 , Nimp=4 ,
    imputationMethod=rep("norm" , 4 ) )
summary(imp.mi1)       # summary of mids.1chain
plot( imp.mi1 ) # trace plot excluding burnin iterations
plot( imp.mi1 , plot.burnin=TRUE ) # trace plot including burnin iterations

# select mids object
imp.mi2 <- imp.mi1$midsobj
summary(imp.mi2) # summary of mids

# apply mice functionality lm.mids
mod <- with( imp.mi2 , lm( bmi ~ age ) )
summary( pool( mod ) )

## Not run:
#############################################################################
# EXAMPLE 2: One chain (mixed data: numeric and factor)
#############################################################################

library(mice)
data(nhanes2, package="mice")
set.seed(9090)

# nhanes2 data in one chain
imp.mi1 <- mice.1chain( nhanes2 , burnin=5 , iter=25 , Nimp=5 )
# summary
summary( imp.mi1$midsobj )

#############################################################################
```

```
# EXAMPLE 3: Multiple imputation with counterfactuals for estimating
#            causal effects (average treatment effects)
# Schafer, J. L., & Kang, J. (2008). Average causal effects from nonrandomized
#    studies: a practical guide and simulated example.
#    Psychological Methods, 13, 279-313.
#############################################################################

data(data.ma01)
dat <- data.ma01[ , 4:11]

# define counterfactuals for reading score for students with and
# without migrational background
dat$read.migrant1 <- ifelse( paste(dat$migrant) == 1 , dat$read , NA )
dat$read.migrant0 <- ifelse( paste(dat$migrant) == 0 , dat$read , NA )

# define imputation method
impmethod <- rep("2l.pls2" , ncol(dat) )
names(impmethod) <- colnames(dat)

# define predictor matrix
pm <- 4*(1 - diag( ncol(dat) ) )    # 4 - use all interactions
rownames(pm) <- colnames(pm) <- colnames(dat)
pm[ c( "read.migrant0" , "read.migrant1") , ] <- 0
# do not use counterfactuals for 'read' as a predictor
pm[ , "read.migrant0"] <- 0
pm[ , "read.migrant1"] <- 0
# define control variables for creation of counterfactuals
pm[ c( "read.migrant0" , "read.migrant1") , c("hisei","paredu","female","books") ] <- 4
  ##   > pm
  ##              math read migrant books hisei paredu female urban read.migrant1 read.migrant0
  ##   math           0    4       4     4     4      4      4     4            0            0
  ##   read           4    0       4     4     4      4      4     4            0            0
  ##   migrant        4    4       0     4     4      4      4     4            0            0
  ##   books          4    4       4     0     4      4      4     4            0            0
  ##   hisei          4    4       4     4     0      4      4     4            0            0
  ##   paredu         4    4       4     4     4      0      4     4            0            0
  ##   female         4    4       4     4     4      4      0     4            0            0
  ##   urban          4    4       4     4     4      4      4     0            0            0
  ##   read.migrant1  0    0       0     4     4      4      4     0            0            0
  ##   read.migrant0  0    0       0     4     4      4      4     0            0            0

# imputation using mice function and PLS imputation with
# predictive mean matching method 'pmm6'
imp <- mice( dat , imputationMethod=impmethod , predictorMatrix=pm ,
            maxit=4 , m=5 , pls.impMethod="pmm5" )

#*** Model 1: Raw score difference
mod1 <- with( imp , lm( read ~ migrant ) )
smod1 <- summary( pool(mod1) )
  ##   > smod1
  ##                 est    se      t     df Pr(>|t|)  lo 95  hi 95 nmis    fmi lambda
  ##   (Intercept) 510.21 1.460 349.37 358.26        0 507.34 513.09   NA 0.1053 0.1004
  ##   migrant     -43.38 3.757 -11.55  62.78        0 -50.89 -35.87  404 0.2726 0.2498
```

```
#*** Model 2: ANCOVA - regression adjustment
mod2 <- with( imp , lm( read ~ migrant + hisei + paredu + female + books) )
smod2 <- summary( pool(mod2) )
  ##   > smod2
  ##                est     se     t     df  Pr(>|t|)    lo 95   hi 95 nmis     fmi   lambda
  ##   (Intercept) 385.1506 4.12027 93.477 3778.66 0.000e+00 377.0725 393.229   NA 0.008678 0.008153
  ##   migrant    -29.1899 3.30263 -8.838   87.46 9.237e-14 -35.7537 -22.626  404 0.228363 0.210917
  ##   hisei        0.9401 0.08749 10.745  160.51 0.000e+00   0.7673   1.113  733 0.164478 0.154132
  ##   paredu       2.9305 0.79081  3.706   41.34 6.190e-04   1.3338   4.527  672 0.339961 0.308780
  ##   female      38.1719 2.26499 16.853 1531.31 0.000e+00  33.7291  42.615    0 0.041093 0.039841
  ##   books       14.0113 0.88953 15.751  154.71 0.000e+00  12.2541  15.768  423 0.167812 0.157123

#*** Model 3a: Estimation using counterfactuals
mod3a <- with( imp , lm( I( read.migrant1 - read.migrant0) ~ 1 ) )
smod3a <- summary( pool(mod3a) )
  ##   > smod3a
  ##                   est    se      t    df Pr(>|t|)  lo 95  hi 95 nmis    fmi lambda
  ##   (Intercept) -22.54 7.498 -3.007 4.315  0.03602 -42.77 -2.311   NA 0.9652 0.9521

#*** Model 3b: Like Model 3a but using student weights
mod3b <- with( imp , lm( I( read.migrant1 - read.migrant0) ~ 1 , weights= data.ma01$studwgt ) )
smod3b <- summary( pool(mod3b) )
  ##   > smod3b
  ##                   est    se      t  df Pr(>|t|)  lo 95  hi 95 nmis    fmi lambda
  ##   (Intercept) -21.88 7.605 -2.877 4.3  0.04142 -42.43 -1.336   NA 0.9662 0.9535

#*** Model 4: Average treatment effect on the treated (ATT, migrants)
#             and non-treated (ATN, non-migrants)
mod4 <- with( imp , lm( I( read.migrant1 - read.migrant0) ~ 0 + as.factor( migrant) )   )
smod4 <- summary( pool(mod4) )
  ##   > smod4
  ##                       est    se      t    df Pr(>|t|)  lo 95   hi 95 nmis    fmi lambda
  ##   as.factor(migrant)0 -23.13 8.664 -2.669  4.27 0.052182 -46.59  0.3416   NA 0.9682 0.9562
  ##   as.factor(migrant)1 -19.95 5.198 -3.837 19.57 0.001063 -30.81 -9.0884   NA 0.4988 0.4501
# ATN = -23.13 and ATT = -19.95

## End(Not run)
```

`mice.impute.2l.contextual.norm`

Imputation by Normal Linear Regression with Contextual Variables

---

Description

This imputation method imputes a variable using linear regression with normally distributed residuals. Including a contextual effects means that an aggregated variable at a cluster level is included as a further covariate.

Usage

```
mice.impute.2l.contextual.norm(y, ry, x, type, ridge = 10^(-5),
   imputationWeights = NULL, interactions = NULL, quadratics = NULL, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE – missing, TRUE – observed)
x	Matrix (n x p) of complete covariates.
type	Type of predictor variables. type=-2 refers to the cluster variable, type=2 denotes a variable for which also a contextual effect is included and type=1 denotes all other variables which are included as 'ordinary' predictors.
ridge	Ridge parameter in the diagonal of $\boldsymbol{X}'\boldsymbol{X}$
imputationWeights	Optional vector of sample weights
interactions	Vector of variable names used for creating interactions
quadratics	Vector of variable names used for creating quadratic terms
...	Further arguments to be passed

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For examples see mice.impute.2l.contextual.pmm.

---

mice.impute.2l.contextual.pmm

Imputation by Predictive Mean Matching with Contextual Variables

---

Description

This imputation method imputes a variable using linear regression with predictive mean matching as the imputation method. Including a contextual effects means that an aggregated variable at a cluster level is included as a further covariate.

Usage

```
mice.impute.2l.contextual.pmm(y, ry, x, type, imputationWeights = NULL,
     interactions = NULL, quadratics = NULL, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE – missing, TRUE – observed)
x	Matrix (n x p) of complete covariates.
type	Type of predictor variables. type=-2 refers to the cluster variable, type=2 denotes a variable for which also a contextual effect is included and type=1 denotes all other variables which are included as 'ordinary' predictors.
imputationWeights	Optional vector of sample weights
interactions	Vector of variable names used for creating interactions
quadratics	Vector of variable names used for creating quadratic terms
...	Further arguments to be passed

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For imputations at level 2 variables see mice::mice.impute.2lonly.norm and mice::mice.impute.2lonly.pmm.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Sequential hierarchical imputation for data.ma05 dataset
#############################################################################

data(data.ma05)
dat <- data.ma05

# empty imputation
imp0 <- mice( dat , m=0 , maxit=0 )
summary(imp0)

# define predictor matrix
predM <- imp0$pred
# exclude student IDs
predM[ , "idstud"] <- 0
# define idclass as the cluster variable (type=-2)
predM[ , "idclass" ] <- -2

# define imputation methods
impMethod <- imp0$method
# initialisiere mit norm
```

```
impMethod <- rep( "norm" , length(impMethod) )
names(impMethod) <- names( imp0$method )
impMethod[ c("idstud","idclass")] <- ""

#*****
# STUDENT LEVEL (Level 1)

# Use a random slope model for Dscore and Mscore as the imputation method.
# Here, variance homogeneity of residuals is assumed (contrary to
# the 2l.norm imputation method in the mice package).
impMethod[ c("Dscore" , "Mscore") ] <- "2l.pan"
predM[ c("Dscore","Mscore") , "misei" ] <- 2 # random slopes on 'misei'
predM[  , "idclass" ] <- -2

# For imputing 'manote' and 'denote' use contextual effects (i.e. cluszer means)
# of variables 'misei' and 'migrant'
impMethod[ c("denote" , "manote") ] <- "2l.contextual.pmm"
predM[ c("denote" , "manote") , c("misei","migrant")] <- 2

# Use no cluster variable 'idclass' for imputation of 'misei'
impMethod[ "misei"] <- "norm"
predM[ "misei" , "idclass"] <- 0 # use no multilevel imputation model

# Variable migrant: contextual effects of Dscore and misei
impMethod[ "migrant"] <- "2l.contextual.pmm"
predM[ "migrant" , c("Dscore" ,  "misei" ) ] <- 2
predM[ "migrant" , "idclass" ] <- -2

#****
# CLASS LEVEL (Level 2)
# impute 'sprengel' and 'groesse' at the level of classes
impMethod[ "sprengel"] <- "2lonly.pmm"
impMethod[ "groesse"] <- "2lonly.norm"
predM[ c("sprengel","groesse") , "idclass" ] <- -2

# do imputation
imp <- mice( dat , predictorMatrix = predM , m = 3 ,  maxit = 4 ,
           imputationMethod = impMethod  , paniter=100)
summary(imp)

## End(Not run)
```

`mice.impute.2l.eap` *Imputation of a Variable with a Known Posterior Distribution*

Description

This function imputes values of a variable for which the mean and the standard deviation of the posterior distribution is known.

Usage

```
mice.impute.2l.eap(y, ry, x, eap, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE – missing, TRUE – observed)
x	Matrix (n x p) of complete covariates.
eap	List with means and standard deviations of the posterior distribution (see Examples). If for multiple variables posterior distributions are known, then it is a list named in which each list entry is named according th variable to be imputed and each list entry contains the variable's EAP and standard deviation of the EAP.
...	Further arguments to be passed

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Imputation based on known posterior distribution
#############################################################################

data(data.ma03)
dat <- data.ma03

# definiere variable 'math_PV' as the plausible value imputation of math
dat$math_PV <- NA
vars <- colnames(dat)
dat1 <- as.matrix( dat[,vars] )

# define imputation methods
impmethod <- rep( "pmm" , length(vars ))
names(impmethod) <- vars
# define plausible value imputation based on EAP and SEEAP for 'math_PV'
impmethod[ "math_PV" ] <- "2l.eap"
eap <- list( "math_PV" = list( "M" = dat$math_EAP  , "SE" = dat$math_SEEAP  ) )
# define predictor matrix
pM <- 1 - diag(1,length(vars))
rownames(pM) <- colnames(pM) <- vars
pM[,c("idstud","math_EAP" , "math_SEEAP") ] <- 0
    # remove some variables from imputation model

# imputation using three parallel chains
```

```
imp1 <- mice( dat1 , m=3 , maxit=5 , imputationMethod=impmethod ,
               predictorMatrix = pM , allow.na =TRUE , eap=eap )
summary(imp1)   # summary

# imputation using one long chain
imp2 <- mice.1chain( dat1 , burnin=10 , iter=20 , Nimp =3 , imputationMethod=impmethod ,
              predictorMatrix = pM ,  allow.na =TRUE , eap=eap )
summary(imp2)    # summary

## End(Not run)
```

`mice.impute.2l.latentgroupmean`
Imputation of Latent and Manifest Group Means for Multilevel Data

---

Description

The imputation method `2l.latentgroupmean` imputes a latent group mean assuming an infinite population of subjects within a group (see Luedtke et al., 2008 or Croon & van Veldhoven, 2007). Therefore, unreliability of group means when treating subjects as indicators is taken into account.

The imputation method `mice.impute.2l.groupmean` just imputes (i.e. computes) the manifest group mean. See also `mice::mice.impute.2lonly.mean`.

The imputation method `mice.impute.2l.groupmean.elim` computes the group mean eliminating the subject under study from the calculation. Therefore, this imputation method will lead to different values of individuals within the same group.

Usage

```
mice.impute.2l.latentgroupmean(y, ry, x, type, pls.facs = NULL,
    imputationWeights = NULL, interactions = NULL, quadratics = NULL, ...)

mice.impute.2l.groupmean(y, ry, x, type, grmeanwarning = TRUE, ...)

mice.impute.2l.groupmean.elim(y, ry, x, type, ...)
```

Arguments

`y`	Incomplete data vector of length `n`
`ry`	Vector of missing data pattern (`FALSE` – missing, `TRUE` – observed)
`x`	Matrix (`n` x `p`) of complete covariates.
`type`	Type of predictor variables. `type=-2` refers to the cluster variable, `type=2` denotes a variable for which also a (latent) group mean should be calculated. Predictors with `type=1` denote all other variables.
`pls.facs`	Number of factors used for PLS regression (optional).
`imputationWeights`	Optional vector of sample weights.

interactions	Vector of variable names used for creating interactions
quadratics	Vector of variable names used for creating quadratic terms
grmeanwarning	An optional logical indicating whether some group means cannot be calculated.
...	Further arguments to be passed.

Details

The imputation of the latent group mean uses the `lmer` function of the **lme4** package. Latent group mean imputation also follows Mislevy (1991).

Value

A vector of length `y` containing imputed group means.

Author(s)

Alexander Robitzsch

References

Croon, M. A., & van Veldhoven, M. J. (2007). Predicting group-level outcome variables from variables measured at the individual level: a latent variable multilevel model. *Psychological Methods*, **12**, 45-57.

Luedtke, O., Marsh, H. W., Robitzsch, A., Trautwein, U., Asparouhov, T., & Muthen, B. (2008). The multilevel latent covariate model: a new, more reliable approach to group-level effects in contextual studies. *Psychological Methods*, **13**, 203-229.

Mislevy, R. J. (1991). Randomization-based inference about latent variables from complex samples. *Psychometrika*, **56**, 177-196.

See Also

`mice::mice.impute.2lonly.mean`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Two-level imputation data.ma05 dataset with imputation
#            of a latent group mean
#############################################################################

data(data.ma05)
dat <- data.ma05

# include manifest group mean for 'Mscore'
dat$M.Mscore <- NA
# include latent group group for 'Mscore'
dat$LM.Mscore <- NA    # => LM: latent group mean
```

```
# empty imputation
imp <- mice( dat , m=0 , maxit=0 )
summary(imp)

# define predictor matrix
predM <- imp$pred
# exclude student ISs
predM[ , "idstud"] <- 0
# idclass is the cluster identifier
predM[ , "idclass" ] <- -2

# define imputation methods
impMethod <- imp$method
# initialize with norm
impMethod <- rep( "norm" , length(impMethod) )
names(impMethod) <- names( imp$method )
impMethod[ c("idstud","idclass")] <- ""

#*****
# STUDENT LEVEL (Level 1)

# Use a random slope model for Dscore and Mscore as the imputation method.
# Here, variance homogeneity of residuals is assumed (contrary to
# the 2l.norm imputation method in the mice package).
impMethod[ c("Dscore" , "Mscore") ] <- "2l.pan"
predM[ c("Dscore","Mscore") , "misei" ] <- 2    # random slopes on 'misei'
predM[  , "idclass" ] <- -2

# For imputing 'manote' and 'denote' use contextual effects (i.e. cluster means)
# of variables 'misei' and 'migrant'
impMethod[ c("denote" , "manote") ] <- "2l.contextual.pmm"
predM[ c("denote" , "manote") , c("misei","migrant")] <- 2

# Use no cluster variable 'idclass' for imputation of 'misei'
impMethod[ "misei"] <- "norm"
predM[ "misei" , "idclass"] <- 0 # use no multilevel imputation model

# Variable migrant: contextual effects of Dscore and misei
impMethod[ "migrant"] <- "2l.contextual.pmm"
predM[ "migrant" , c("Dscore" ,  "misei" ) ] <- 2
predM[ "migrant" , "idclass" ] <- -2

#****
# CLASS LEVEL (Level 2)
# impute 'sprengel' and 'groesse' at the level of classes
impMethod[ "sprengel"] <- "2lonly.pmm2"
impMethod[ "groesse"] <- "2lonly.norm2"
predM[ c("sprengel","groesse") , "idclass" ] <- -2

# manifest group mean for Mscore
impMethod[ "M.Mscore" ] <- "2l.groupmean"
# latent group mean for Mscore
impMethod[ "LM.Mscore" ] <- "2l.latentgroupmean"
```

```
predM[ "M.Mscore" , "Mscore" ] <- 2

# covariates for latent group mean of 'Mscore'
predM[ "LM.Mscore" , "Mscore" ] <- 2
predM[ "LM.Mscore" , c( "Dscore" , "sprengel" ) ] <- 1

# do imputations
imp <- mice( dat , predictorMatrix = predM , m =3 ,  maxit = 4 ,
         imputationMethod = impMethod  , allow.na = TRUE , pan.iter=100)

## End(Not run)
```

`mice.impute.2l.plausible.values` *Plausible Value Imputation using Classical Test Theory and Based on Individual Likelihood*

Description

This imputation function performs unidimensional plausible value imputation if (subject-wise) measurement errors or the reliability of the scale is known (Mislevy, 1991; see also Asparouhov & Muthen, 2010; Blackwell, Honaker & King, 2011). The function also allows the input of an individual likelihood obtained by fitting an item response model.

Usage

```
mice.impute.2l.plausible.values(y, ry, x, type, alpha = NULL,
    alpha.se = 0, scale.values = NULL, sig.e.miss = 1e+06,
    like=NULL , theta=NULL , normal.approx=NULL ,
   pviter = 15, imputationWeights = rep(1, length(y)), plausible.value.print = TRUE,
    pls.facs = NULL, interactions = NULL,
    quadratics = NULL, ...)
```

Arguments

`y`	Incomplete data vector of length `n`
`ry`	Vector of missing data pattern (`FALSE` – missing, `TRUE` – observed)
`x`	Matrix (`n` $\times$ `p`) of complete covariates.
`type`	Type of predictor variables. `type=3` refers to items belonging to a scale to be imputed. A cluster (grouping) variable is defined by `type=-2`. If for some predictors, the cluster means should also be included as predictors, then specify `type=2` (see Imputation Model 3 of Example 1).
`alpha`	A known reliability estimate. An optional standard error of the estimate can be provided in `alpha.se`
`alpha.se`	Optional numeric value of the standard error of the `alpha` reliability estimate if in every iteration a new reliability should be sampled.

`scale.values`	A list consisting of scale values of scale values and its corresponding standard errors (see Example 1).
`sig.e.miss`	A standard error of measurement for cases with missing values on a scale
`like`	Individual likelihood evaluated at `theta`
`theta`	Grid of unidimensional latent variable
`normal.approx`	Logical indicating whether the individual posterior should be approximated by a normal distribution
`pviter`	Number of iterations in each imputation which should be run until the plausible values are drawn
`imputationWeights`	Optional vector of sample weights
`plausible.value.print`	An optional logical indicating whether some information about the plausible value imputation should be printed at the console
`pls.facs`	Number of PLS factors if PLS dimension reduction is used
`interactions`	Vector of variable names used for creating interactions
`quadratics`	Vector of variable names used for creating quadratic terms
`...`	Further objects to be passed

Details

The linear model is assumed for drawing plausible values of a variable Y contaminated by measurement error. Assuming $Y = \theta + e$ and a linear regression model for θ

$$\theta = \boldsymbol{X}\beta + \epsilon$$

(plausible value) imputations from the posterior distribution $P(\theta|Y, \boldsymbol{X})$ are drawn. See Mislevy (1991) for details.

Value

A vector of length `nrow(x)` containing imputed plausible values.

Note

Plausible value imputation is also known as multiple overimputation (Blackwell, Honaker & King, 2015a, 2015b) which is implemented in the **Amelia** package, see `Amelia::moPrep` and `Amelia::amelia`.

Author(s)

Alexander Robitzsch

References

Asparouhov, T., & Muthen, B. (2010). *Plausible values for latent variables using Mplus*. Technical Report. https://www.statmodel.com/papers.shtml

Blackwell, M., Honaker, J., & King, G. (2011). *Multiple overimputation: A unified approach to measurement error and missing data*. Technical Report.

Blackwell, M., Honaker, J., & King, G. (2015a). A unified approach to measurement error and missing data: Overview and applications. *Sociological Methods & Research*, **xx**, xxx-xxx.

Blackwell, M., Honaker, J., & King, G. (2015b). A unified approach to measurement error and missing data: Details and extensions. *Sociological Methods & Research*, **xx**, xxx-xxx.

Mislevy, R. J. (1991). Randomization-based inference about latent variables from complex samples. *Psychometrika*, **56**, 177-196.

See Also

See TAM::tam.latreg for fitting latent regression models.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Plausible value imputation for data.ma04 | 2 scales
#############################################################################

data(data.ma04)
dat <- data.ma04

# Scale 1 consists of items A1,...,A4
# Scale 2 consists of items B1,...,B5
dat$scale1 <- NA
dat$scale2 <- NA

# empty imputation
imp <- mice( dat , m=0 , maxit=0 )
summary(imp)

# define predictors
predM <- imp$pred
# define imputation methods
impMethod <- imp$method
impMethod <- rep( "norm" , length(impMethod) )
names(impMethod) <- names( imp$method )

# look at missing proportions
colSums( is.na(dat) )

# redefine imputation methods for plausible value imputation
impMethod[ "scale1" ] <- "2l.plausible.values"
predM[ "scale1" ,  ] <- 1
predM[ "scale1" , c("A1" , "A2" ,  "A3" , "A4" ) ] <- 3
    # items corresponding to a scale should be declared by a 3 in the predictor matrix
```

```
impMethod[ "scale2" ] <- "2l.plausible.values"
predM[ ,"scale2"  ] <- 0
predM[ "scale2" ,  c("A2","A3","A4","V6","V7") ] <- 1
diag(predM) <- 0

# use imputed scale values as predictors for V5, V6 and V7
predM[ c("V5","V6","V7") , c("scale1","scale2" ) ] <- 1
# exclude for V5, V6 and V7 the items of scales A and B as predictors
predM[ c("V5","V6","V7") , c( paste0("A",2:4) , paste0("B",1:5) ) ] <- 0
# exclude 'group' as a predictor
predM[,"group"] <- 0

# look at imputation method and predictor matrix
impMethod
predM

#-------------------------------
# Parameter for imputation
#***
# scale 1 (A1,...,A4)
# known Cronbach's Alpha
alpha <- NULL
alpha <- list( "scale1" = .8 )
alpha.se <- list( "scale1" = .05 )  # sample alpha with a standard deviation of .05

#***
# scale 2 (B1,...,B5)
# means and SE's of scale scores are assumed to be known
M.scale2 <- rowMeans( dat[  , paste("B",1:5,sep="")  ] )
# M.scale2[ is.na( m1) ] <- mean( M.scale2 , na.rm=TRUE )
SE.scale2 <- rep( sqrt( var(M.scale2,na.rm=T)*(1-.8) ) , nrow(dat) )
# => heterogeneous measurement errors are allowed
scale.values <- list( "scale2" = list( "M" = M.scale2 , "SE" = SE.scale2 ) )

#*** Imputation Model 1: Imputation four using parallel chains
imp1 <- mice( dat , predictorMatrix = predM , m = 4, maxit = 5 ,
        alpha.se = alpha.se ,  imputationMethod = impMethod ,  allow.na = TRUE  , alpha = alpha,
          scale.values = scale.values  )
summary(imp1)

# extract first imputed dataset
dat11 <- complete( imp , 1 )

#*** Imputation Model 2: Imputation using one long chain
imp2 <- mice.1chain( dat , predictorMatrix = predM , burnin=10 , iter=20 , Nimp=4 ,
        alpha.se = alpha.se ,  imputationMethod = impMethod ,  allow.na = TRUE  , alpha = alpha,
          scale.values = scale.values  )
summary(imp2)

#-------------
#*** Imputation Model 3: Imputation including  group level variables

# use group indicator for plausible value estimation
```

```
predM[ "scale1" , "group" ] <- -2
# V7 and B1 should be aggregated at the group level
predM[ "scale1" , c("V7","B1") ] <- 2
predM[ "scale2" , "group" ] <- -2
predM[ "scale2" , c("V7","A1") ] <- 2

# perform single imputation (m=1)
imp <- mice( dat , predictorMatrix = predM , m = 1 , maxit=10 ,
            imputationMethod = impMethod ,  allow.na = TRUE  , alpha = alpha,
            scale.values = scale.values )
dat10 <- complete(imp)

# multilevel model
library(lme4)
mod <- lmer( scale1 ~ ( 1 | group) , data = dat11 )
summary(mod)

mod <- lmer( scale1 ~ ( 1 | group) , data = dat10)
summary(mod)

#############################################################################
# SIMULATED EXAMPLE 2: Plausible value imputation with chained equations
#############################################################################

# - simulate a latent variable theta and dichotomous item responses
# - two covariates X in which the second covariate has measurement error

library(sirt)
library(TAM)
library(lavaan)

set.seed(7756)
N <- 2000    # number of persons
I <- 10     # number of items

# simulate covariates
X <- mvrnorm( N , mu=c(0,0) , Sigma = matrix( c(1,.5,.5,1) ,2 ,2 ) )
colnames(X) <- paste0("X",1:2)
# second covariate with measurement error with variance var.err
var.err <- .3
X.err <- X
X.err[,2] <-X[,2] + rnorm(N, sd = sqrt(var.err) )
# simulate theta
theta <- .5*X[,1] + .4*X[,2] + rnorm( N , sd = .5 )
# simulate item responses
itemdiff <- seq( -2 , 2 , length=I)  # item difficulties
dat <- sirt::sim.raschtype( theta , b = itemdiff )

#***********************
#*** Model 0: Regression model with true variables
mod0 <- lm( theta ~ X )
summary(mod0)
```

```
#**********************
# plausible value imputation for abilities and error-prone
# covariates using the mice package

# creating the likelihood for plausible value for abilities
mod11 <- TAM::tam.mml( dat )
likePV <- IRT.likelihood(mod11)
# creating the likelihood for error-prone covariate X2
# The known measurement error variance is 0.3.
lavmodel <- "
  X2true =~ 1*X2
  X2 ~~ 0.3*X2
    "
mod12 <- lavaan::cfa( lavmodel , data = as.data.frame(X.err) )
summary(mod12)
likeX2 <- IRTLikelihood.cfa( data= X.err , cfaobj=mod12)
str(likeX2)

#-- create data input for mice package
data <- data.frame( "PVA" = NA , "X1" = X[,1] , "X2" = NA  )
vars <- colnames(data)
V <- length(vars)
predictorMatrix <- 1 - diag(V)
rownames(predictorMatrix) <- colnames(predictorMatrix) <- vars
imputationMethod <- rep("norm" , V )
names(imputationMethod) <- vars
imputationMethod[c("PVA","X2")] <- "2l.plausible.values"

#-- create argument lists for plausible value imputation
# likelihood and theta grid of plausible value derived from IRT model
like <- list( "PVA" = likePV  , "X2" = likeX2 )
theta <- list( "PVA" = attr(likePV,"theta") ,
                "X2" = attr(likeX2 , "theta") )
#-- initial imputations
data.init <- data
data.init$PVA <- mod11$person$EAP
data.init$X2 <- X.err[,"X2"]

#-- imputation using the mice and miceadds package
imp1 <- mice::mice( as.matrix(data) , predictorMatrix = predictorMatrix , m = 4, maxit = 6 ,
            imputationMethod = imputationMethod ,  allow.na = TRUE ,
            theta=theta , like=like , data.init=data.init )
summary(imp1)

# compute linear regression
mod4a <- with( imp1 , lm( PVA ~ X1 + X2 ) )
summary( pool(mod4a) )

## End(Not run)
```

`mice.impute.2l.pls2` *Imputation using Partial Least Squares for Dimension Reduction*

Description

This function imputes a variable with missing values using PLS regression (Mevik & Wehrens, 2007) for a dimension reduction of the predictor space.

Usage

```
mice.impute.2l.pls2(y, ry, x, type, pls.facs = NULL,
   pls.impMethod = "pmm", pls.print.progress = TRUE,
   imputationWeights = rep(1, length(y)), pcamaxcols = 1E+09,
   tricube.pmm.scale = NULL, min.int.cor = 0, min.all.cor=0,
   N.largest = 0, pls.title = NULL, print.dims = TRUE,
   pls.maxcols=5000 , ...)

mice.impute.2l.pls(y, ry, x, type, pls.facs = NULL,
   pls.impMethod = "tricube.pmm2", pls.method = NULL,
   pls.print.progress = TRUE, imputationWeights = rep(1, length(y)),
   pcamaxcols = 1E+09, tricube.pmm.scale = NULL, min.int.cor = 0, min.all.cor=0,
   N.largest = 0, pls.title = NULL, print.dims = TRUE, ...)
```

Arguments

`y`	Incomplete data vector of length `n`
`ry`	Vector of missing data pattern (`FALSE` – missing, `TRUE` – observed)
`x`	Matrix (`n` x `p`) of complete covariates.
`type`	`type=1` – variable is used as a predictor, `type=4` – create interactions with the specified variable with all other predictors, `type=5` – create a quadratic term of the specified variable `type=6` – if some interactions are specified, ignore the variables with entry `6` when creating interactions `type=-2` – specification of a cluster variable. The cluster mean of the outcome `y` (when eliminating the subject under study) is included as a further predictor in the imputation.
`pls.facs`	Number of factors used in PLS regression. This argument can also be specified as a list defining different numbers of factors for all variables to be imputed.
`pls.impMethod`	Imputation method based in the PLS regression model: `norm` – normal linear regression `pmm` – predictive mean matching (`pmm` method from **mice**) `pmm5` – predictive mean matching (`pmm5` method from **miceadds**) `tricube.pmm`/`tricube.pmm2` – predictive mean matching with tricube kernel `xplsfacs` – create only PLS factors of the regression model
`pls.method`	Calculation method of PLS regression. See `pls::plsr` (**pls**) for more details.
`pls.print.progress`	Print progress during PLS regression.

`imputationWeights`
: Vector of sample weights to be used in imputation models.

`pcamaxcols`
: Maximum number of principal components.

`tricube.pmm.scale`
: Scale factor for tricube predictive mean matching.

`min.int.cor`
: Minimum absolute correlation for an interaction of two predictors to be included in the PLS regression model

`min.all.cor`
: Minimum absolute correlation for inclusion in the PLS regression model.

`N.largest`
: Number of variable to be included which do have the largest absolute correlations.

`pls.title`
: Title for progress print in console output.

`print.dims`
: An optional logical indicating whether dimensions of inputs should be printed.

`pls.maxcols`
: Maximum number of interactions to be created.

`...`
: Further arguments to be passed.

Details

The function `mice.impute.2l.pls2` uses `kernelpls.fit2` instead of `kernelpls.fit` from the **pls** package and is a bit faster.

Value

A vector of length `nmis=sum(!ry)` with imputations if `pls.impMethod != "xplsfacs"`. In case of `pls.impMethod == "xplsfacs"` a matrix with PLS factors is computed.

Author(s)

Alexander Robitzsch

References

Mevik, B. H., & Wehrens, R. (2007). The **pls** package: Principal component and partial least squares regression in R. *Journal of Statistical Software*, **18**, 1-24.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: PLS imputation method for internet data
#############################################################################

data(data.internet)
dat <- data.internet

# specify predictor matrix
predictorMatrix <- matrix( 1 , ncol(dat) , ncol(dat) )
rownames(predictorMatrix) <- colnames(predictorMatrix) <- colnames(dat)
diag( predictorMatrix) <- 0
```

```
# use PLS imputation method for all variables
impMethod <- rep( "2l.pls2" , ncol(dat) )
names(impMethod) <- colnames(dat)

# define predictors for interactions (entries with type 4 in predictorMatrix)
predictorMatrix[c("IN1","IN15","IN16"),c("IN1","IN3","IN10","IN13")] <- 4
# define predictors which should appear as linear and quadratic terms (type 5)
predictorMatrix[c("IN1","IN8","IN9","IN10","IN11"),c("IN1","IN2","IN7","IN5")] <- 5

# use 9 PLS factors for all variables
pls.facs <- as.list( rep( 9 , length(impMethod) ) )
names(pls.facs) <- names(impMethod)
pls.facs$IN1 <- 15   # use 15 PLS factors for variable IN1

# choose norm or pmm imputation method
pls.impMethod <- as.list( rep("norm" , length(impMethod) ) )
names(pls.impMethod) <- names(impMethod)
pls.impMethod[ c("IN1","IN6")] <- "pmm5"

# Model 1: Three parallel chains
imp1 <- mice(data = dat , imputationMethod = impMethod ,
     m=3 , maxit=5 , predictorMatrix = predictorMatrix ,
     pls.facs = pls.facs , # number of PLS factors
     pls.impMethod = pls.impMethod ,  # Imputation Method in PLS imputation
     pls.print.progress = TRUE )
summary(imp1)

# Model 2: One long chain
imp2 <- mice.1chain(data = dat , imputationMethod = impMethod ,
     burnin=10 , iter=21 , Nimp=3 , predictorMatrix = predictorMatrix ,
     pls.facs = pls.facs , pls.impMethod = pls.impMethod )
summary(imp2)

## End(Not run)
```

`mice.impute.2lonly.pmm2`

Imputation at Level 2 (in **miceadds***)*

Description

These functions are just bug fixes of `mice.impute.2lonly.pmm` and `mice.impute.2lonly.norm` in **mice** (version 2.21).

Usage

```
mice.impute.2lonly.pmm2(y, ry, x, type, ...)
mice.impute.2lonly.norm2(y, ry, x, type, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE=missing, TRUE=observed)
x	Matrix (n x p) of complete covariates. Only numeric variables are permitted for usage of this function.
type	Group identifier must be specified by '-2'. Predictors must be specified by '1'.
...	Other named arguments.

Value

A vector of length nmis with imputations.

Author(s)

Alexander Robitzsch

See Also

mice.impute.2lonly.pmm (**mice**), mice.impute.2lonly.norm (**mice**)

---

mice.impute.grouped *Imputation of a Variable with Grouped Values*

Description

Imputes a variable with continuous values whose original values are only available as grouped values.

Usage

```
mice.impute.grouped(y, ry, x, low=NULL , upp=NULL ,  ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE – missing, TRUE – observed)
x	Matrix (n x p) of complete covariates.
low	Vector with lower bound of grouping interval
upp	Vector with upper bound of grouping interval
...	Further arguments to be passed

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

See Also

This function uses the **grouped** package.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Imputation of grouped data
#############################################################################

data(data.ma06)
data <- data.ma06

# define the variable "FC_imp" which should contain the variables to be imputed
data$FC_imp <- NA
V <- ncol(data)
# variables not to be used for imputation
vars_elim <-  c("id" , "FC","FC_low","FC_upp")

# define imputation methods
impM <- rep("norm" , V)
names(impM) <- colnames(data)
impM[  vars_elim ] <- ""
impM[ "FC_imp" ] <- "grouped"

# define predictor matrix
predM <- 1 - diag( 0 , V)
rownames(predM) <- colnames(predM) <- colnames(data)
predM[vars_elim, ] <- 0
predM[,vars_elim] <- 0

# define lower and upper boundaries of the grouping intervals
low <- list("FC_imp" = data$FC_low )
upp <- list("FC_imp" = data$FC_upp )

# perform imputation
imp <- mice( data , imputationMethod = impM , predictorMatrix = predM ,
    m=1 , maxit=3 , allow.na=TRUE ,  low=low , upp=upp)
head( complete(imp ))

## End(Not run)
```

`mice.impute.pmm3` *Imputation by Predictive Mean Matching (in* **miceadds***)*

Description

This function imputes values by predictive mean matching like the `mice::mice.impute.pmm` method in the **mice** package.

Usage

```
mice.impute.pmm3(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)
mice.impute.pmm4(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)
mice.impute.pmm5(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)
mice.impute.pmm6(y, ry, x, donors = 3, noise = 10^5, ridge = 10^(-5), ...)
```

Arguments

`y`	Incomplete data vector of length `n`
`ry`	Vector of missing data pattern (`FALSE` – missing, `TRUE` – observed)
`x`	Matrix (`n` x `p`) of complete covariates.
`donors`	Number of donors used for imputation
`noise`	Numerical value to break ties
`ridge`	Ridge parameter in the diagonal of $\boldsymbol{X}'\boldsymbol{X}$
`...`	Further arguments to be passed

Details

The imputation method `pmm3` imitates `mice::mice.impute.pmm` imputation method in **mice**.

The imputation method `pmm4` ignores ties in predicted y values. With many predictors, this does not probably implies any substantial problem.

The imputation method `pmm5` suffers from the same problem. Contrary to the other PMM methods, it searches D donors (specified by `donors`) smaller than the predicted value and D donors larger than the predicted value and randomly samples a value from this set of $2 \cdot D$ donors.

The imputation method `pmm6` is just the **Rcpp** implementation of `pmm5`.

Value

A vector of length `nmis=sum(!ry)` with imputed values.

Author(s)

Alexander Robitzsch

See Also

See `data.largescale` and `data.smallscale` for speed comparisons of different functions for predictive mean matching.

Examples

```
## Not run:
#############################################################################
# SIMULATED EXAMPLE 1: Two variables x and y with missing y
#############################################################################
set.seed(1413)

rho <- .6   # correlation between x and y
N <- 6800    # number of cases
x <- rnorm(N)
My <- .35   # mean of y
y.com <- y <- My + rho * x + rnorm(N , sd = sqrt( 1 - rho^2 ) )

# create missingness on y depending on rho.MAR parameter
rho.mar <- .4    # correlation response tendency z and x
missrate <- .25  # missing response rate
# simulate response tendency z and missings on y
z <- rho.mar * x + rnorm(N , sd = sqrt( 1 - rho.mar^2 ) )
y[ z < qnorm( missrate ) ] <- NA
dat <- data.frame(x , y )

# mice imputation
impmethod <- rep("pmm" , 2 )
names(impmethod) <- colnames(dat)

# pmm (in mice)
imp1 <- mice( as.matrix(dat) , m=1 , maxit=1 , imputationMethod=impmethod)
# pmm3 (in miceadds)
imp3 <- mice( as.matrix(dat) , m=1 , maxit=1 ,
        imputationMethod=gsub("pmm","pmm3" ,impmethod)  )
# pmm4 (in miceadds)
imp4 <- mice( as.matrix(dat) , m=1 , maxit=1 ,
        imputationMethod=gsub("pmm","pmm4" ,impmethod)  )
# pmm5 (in miceadds)
imp5 <- mice( as.matrix(dat) , m=1 , maxit=1 ,
        imputationMethod=gsub("pmm","pmm5" ,impmethod)  )
# pmm6 (in miceadds)
imp6 <- mice( as.matrix(dat) , m=1 , maxit=1 ,
        imputationMethod=gsub("pmm","pmm6" ,impmethod)  )

dat.imp1 <- complete( imp1 , 1 )
dat.imp3 <- complete( imp3 , 1 )
dat.imp4 <- complete( imp4 , 1 )
dat.imp5 <- complete( imp5 , 1 )
dat.imp6 <- complete( imp6 , 1 )

dfr <- NULL
# means
dfr <- rbind( dfr , c( mean( y.com ) , mean( y , na.rm=TRUE ) , mean( dat.imp1$y)  ,
    mean( dat.imp3$y) , mean( dat.imp4$y)  , mean( dat.imp5$y)  ,  mean( dat.imp6$y)  ) )
# SD
dfr <- rbind( dfr , c( sd( y.com ) , sd( y , na.rm=TRUE ) , sd( dat.imp1$y)  ,
```

```
    sd( dat.imp3$y) , sd( dat.imp4$y)  , sd( dat.imp5$y)  , sd( dat.imp6$y) ) )
# correlations
dfr <- rbind( dfr , c( cor( x,y.com ) , cor( x[ ! is.na(y) ] , y[ ! is.na(y) ] ) ,
    cor( dat.imp1$x , dat.imp1$y) , cor( dat.imp3$x , dat.imp3$y) ,
    cor( dat.imp4$x , dat.imp4$y) , cor( dat.imp5$x , dat.imp5$y)  ,
    cor( dat.imp6$x , dat.imp6$y)
        ) )
rownames(dfr) <- c("M_y" , "SD_y" , "cor_xy" )
colnames(dfr) <- c("compl" , "ld" , "pmm" , "pmm3" , "pmm4" , "pmm5","pmm6")
##           compl     ld    pmm   pmm3   pmm4   pmm5   pmm6
##   M_y    0.3306 0.4282 0.3314 0.3228 0.3223 0.3264 0.3310
##   SD_y   0.9910 0.9801 0.9873 0.9887 0.9891 0.9882 0.9877
##   cor_xy 0.6057 0.5950 0.6072 0.6021 0.6100 0.6057 0.6069

## End(Not run)
```

`mice.impute.tricube.pmm`

Imputation by Tricube Predictive Mean Matching

Description

This function performs tricube predictive mean matching (see http://www.rdocumentation.org/packages/Hmisc/functions/aregImpute) in which donors are weighted according to distances of predicted values.

Usage

```
mice.impute.tricube.pmm(y, ry, x, tricube.pmm.scale = 0.2, tricube.boot = FALSE, ...)

mice.impute.tricube.pmm2(y, ry, x, tricube.pmm.scale = 0.2, tricube.boot = FALSE, ...)
```

Arguments

`y`	Incomplete data vector of length `n`
`ry`	Vector of missing data pattern (`FALSE` – missing, `TRUE` – observed)
`x`	Matrix (`n` x `p`) of complete covariates.
`tricube.pmm.scale`	A scaling factor for traicube matching. The default is 0.2.
`tricube.boot`	A logical indicating whether tricube matching should be performed using a bootstrap sample
`...`	Further arguments to be passed

Value

A vector of length `nmis=sum(!ry)` with imputed values.

Note

The imputation method `tricube.pmm2` is ordinarily somewhat faster than `tricube.pmm`.

Author(s)

Alexander Robitzsch

See Also

http://www.rdocumentation.org/packages/Hmisc/functions/aregImpute

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Tricube predictive mean matching for nhanes data
#############################################################################

library(mice)
data(nhanes, package="mice")
set.seed(9090)

#*** Model 1: Use default of tricube predictive mean matching
varnames <- colnames(nhanes)
VV <- length(varnames)
imputationMethod <- rep("tricube.pmm2" , VV )
names(imputationMethod) <- varnames
# imputation with mice
imp.mi1 <- mice( nhanes , m=5 , maxit=4 , imputationMethod= imputationMethod )

#*** Model 2: use item-specific imputation methods
iM2 <- imputationMethod
iM2["bmi"] <- "pmm6"
# use tricube.pmm2 for hyp and chl
# select different scale parameters for these variables
tricube.pmm.scale1 <- list( "hyp" = .15 , "chl" = .30 )
imp.mi2 <- mice.1chain( nhanes , burnin=5 , iter=20 , Nimp=4 ,
    imputationMethod= iM2 , tricube.pmm.scale=tricube.pmm.scale1  )

## End(Not run)
```

---

mice.impute.weighted.norm

Imputation by a Weighted Linear Normal Regression

Description

Imputation by a weighted linear normal regression.

Usage

```
mice.impute.weighted.norm(y, ry, x, ridge = 1e-05, pls.facs = NULL,
     imputationWeights = NULL, interactions = NULL, quadratics = NULL, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE – missing, TRUE – observed)
x	Matrix (n x p) of complete covariates.
ridge	Ridge parameter in the diagonal of $\boldsymbol{X}'\boldsymbol{X}$
imputationWeights	Optional vector of sampling weights
pls.facs	Number of factors in PLS regression (if used). The default is NULL which means that no PLS regression is used for dimension reduction.
interactions	Optional vector of variables for which interactions should be created
quadratics	Optional vector of variables which should also be included as quadratic effects.
...	Further arguments to be passed

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For examples see mice.impute.weighted.pmm

---

mice.impute.weighted.pmm

Imputation by Weighted Predictive Mean Matching

Description

Imputation by predictive mean matching using sampling weights.

Usage

```
mice.impute.weighted.pmm(y, ry, x, imputationWeights = NULL,
      pls.facs = NULL, interactions = NULL, quadratics = NULL, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (FALSE – missing, TRUE – observed)
x	Matrix (n x p) of complete covariates.
imputationWeights	Optional vector of sampling weights
pls.facs	Number of factors in PLS regression (if used). The default is NULL which means that no PLS regression is used for dimension reduction.
interactions	Optional vector of variables for which interactions should be created
quadratics	Optional vector of variables which should also be included as quadratic effects.
...	Further arguments to be passed

Value

A vector of length nmis=sum(!ry) with imputed values.

Author(s)

Alexander Robitzsch

See Also

For imputation with the linear normal regression and sampling weights see mice.impute.weighted.norm.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Imputation using sample weights
#############################################################################

data( data.ma01)
set.seed(977)

# select subsample
dat <- as.matrix(data.ma01)
dat <- dat[ 1:1000 , ]

# empty imputation
imp0 <- mice( dat , m=0 , maxit=0)

# redefine imputation methods
meth <- imp0$method
meth[ meth == "pmm"  ] <- "weighted.pmm"
meth[ c("paredu" , "books" , "migrant" ) ] <- "weighted.norm"
# redefine predictor matrix
pm <- imp0$predictorMatrix
pm[ , 1:3 ] <- 0
# do imputation
```

```
imp <- mice( dat , predictorMatrix=pm , imputationMethod=meth ,
    imputationWeights= dat[,"studwgt"] , m=3 , maxit=5)

## End(Not run)
```

`mice.nmi` *Nested Multiple Imputation*

Description

Performs nested multiple imputation (Rubin, 2003) for the functions `mice::mice` and `mice.1chain`. The function `mice.nmi` generates an object of class `mids.nmi`.

Usage

```
mice.nmi(datlist, type = "mice", ...)

## S3 method for class 'mids.nmi'
summary(object, ...)
```

Arguments

`datlist`	List of datasets for which nested multiple imputation should be applied
`type`	Imputation model: `type="mice"` for `mice::mice` or `type="mice.1chain"` for `mice.1chain`.
`...`	Arguments to be passed to `mice::mice` or `mice.1chain`.
`object`	Object of class `mids.nmi`.

Value

Object of class `mids.nmi` with entries

`imp`	List of nested multiply imputed datasets whose entries are of class `mids` or `mids.1chain`.
`Nimp`	Number of between and within imputations.

Author(s)

Alexander Robitzsch

References

Rubin, D. B. (2003). Nested multiple imputation of NMES via partially incompatible MCMC. *Statistica Neerlandica*, **57(1)**, 3-18.

See Also

For imputation models see `mice::mice` and `mice.1chain`.

Functions for analysis for nested multiply imputed datasets: `complete.mids.nmi`, `with.mids.nmi`, `pool.mids.nmi`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Nested multiple imputation for TIMSS data
#############################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2
   # list of 5 datasets containing 5 plausible values

#** define imputation method and predictor matrix
data <- datlist[[1]]
V <- ncol(data)
# variables
vars <- colnames(data)
# variables not used for imputation
vars_unused <- scan.vec("IDSTUD TOTWGT  JKZONE  JKREP" )

#- define imputation method
impMethod <- rep("norm" , V )
names(impMethod) <- vars
impMethod[ vars_unused ] <- ""

#- define predictor matrix
predM <- matrix( 1 , V , V )
colnames(predM) <- rownames(predM) <- vars
diag(predM) <- 0
predM[ , vars_unused ] <- 0

#***************
# (1) nested multiple imputation using mice
imp1 <- mice.nmi( datlist , imputationMethod=impMethod , predictorMatrix=predM,
                m=4 , maxit=3 )
summary(imp1)

#***************
# (2) nested multiple imputation using mice.1chain
imp2 <- mice.nmi( datlist , imputationMethod=impMethod , predictorMatrix=predM ,
            Nimp=4 , burnin=10 ,iter =22, type="mice.1chain")
summary(imp2)

## End(Not run)
```

`micombine.chisquare` *Combination of Chi Square Statistics of Multiply Imputed Datasets*

Description

This function does inference for the χ^2 statistic based on multiply imputed datasets (see e.g. Enders, 2010, p. 239 ff.; Allison, 2002). This function is also denoted as the D_2 statistic.

Usage

```
micombine.chisquare(dk, df, display = TRUE)
```

Arguments

`dk`	Vector of chi square statistics
`df`	Degrees of freedom of χ^2 statistic
`display`	An optional logical indicating whether results should be printed at the R console.

Value

A vector with following entries

`D`	Combined D_2 statistic which is approximately F-distributed with (`df`, `df2`) degrees of freedom
`p`	The p value corresponding to `D`
`df`	Denominator degrees of freedom
`df2`	Numerator degrees of freedom
`chisq.approx`	Chi square approximation of the D_2 statistic
`p.approx`	The p value corresponding to the D_2 statistic

Author(s)

Alexander Robitzsch

References

Allison, P. D. (2002). *Missing data*. Newbury Park, CA: Sage.

Enders, C. K. (2010). *Applied missing data analysis*. Guilford Press.

See Also

See also `mice::pool.compare` for a Wald test to compare two fitted models in the **mice** package.

Examples

```
#############################################################################
# EXAMPLE 1: Chi square values of analyses from 7 multiply imputed datasets
#############################################################################

# Vector of 7 chi square statistics
dk <- c(24.957,18.051,18.812,17.362,21.234,18.615,19.84)
dk.comb <- micombine.chisquare(dk=dk, df=4 )
##   Combination of Chi Square Statistics for Multiply Imputed Data
##   Using 7 Imputed Data Sets
##   F(4,594.01)=4.486     p=0.00141
##   Chi Square Approximation Chi2(4)=17.946     p=0.00126
```

`micombine.cor` *Combination of Correlations for Multiply Imputed Data Sets*

Description

Statistical inference for correlation coefficients for multiply imputed datasets

Usage

```
micombine.cor(mi.res, variables = 1:(ncol(mi.list[[1]])), conf.level = 0.95,
     method="pearson")
```

Arguments

`mi.res`	Object of class `mids` or `mids.1chain`
`variables`	Indices of variables for selection
`conf.level`	Confidence level
`method`	Method for calculating correlations. Must be one of `"pearson"` or `"spearman"`. The default is the calculation of the Pearson correlation.

Value

A data frame containing the correlation coefficient (`r`) and its corresponding standard error (`rse`), fraction of missing information (`fmi`) and a t value (`t`).

Author(s)

Alexander Robitzsch

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: nhanes data | combination of correlation coefficients
#############################################################################

library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes data in one chain
imp.mi <- mice.1chain( nhanes , burnin=5 , iter=20 , Nimp=4 ,
        imputationMethod=rep("norm" , 4 ) )
# correlation coefficients of variables 4,2 and 3 (indexed in nhanes data)
res <- micombine.cor(mi.res=imp.mi, variables = c(4,2,3) )
  ##     variable1 variable2       r    rse fisher_r fisher_rse    fmi       t      p
  ##   1       chl       bmi  0.2458 0.2236   0.2510     0.2540 0.3246  0.9879 0.3232
  ##   2       chl       hyp  0.2286 0.2152   0.2327     0.2413 0.2377  0.9643 0.3349
  ##   3       bmi       hyp -0.0084 0.2198  -0.0084     0.2351 0.1904 -0.0358 0.9714
  ##     lower95 upper95
  ##   1 -0.2421  0.6345
  ##   2 -0.2358  0.6080
  ##   3 -0.4376  0.4239

#############################################################################
# EXAMPLE 2: nhanes data | comparing different correlation coefficients
#############################################################################

library(psych)
library(mitools)

# imputing data
imp1 <- mice( nhanes ,  imputationMethod=rep("norm" , 4 ) )
summary(imp1)

#*** Pearson correlation
res1 <- micombine.cor(mi.res=imp1, variables = c(4,2) )

#*** Spearman rank correlation
res2 <- micombine.cor(mi.res=imp1, variables = c(4,2) ,  method="spearman")

#*** Kendalls tau
# test of computation of tau for first imputed dataset
dat1 <- complete(imp1, action=1)
tau1 <- psych::corr.test(x=dat1[,c(4,2)], method = "kendall")
tau1$r[1,2]    # estimate
tau1$se     # standard error

# results of Kendalls tau for all imputed datasets
res3 <- with( data=imp1 ,
        expr = psych::corr.test( x = cbind( chl , bmi ) , method="kendall") )
# extract estimates
```

```
betas <- lapply( res3$analyses , FUN = function(ll){ ll$r[1,2] } )
# extract variances
vars <- lapply( res3$analyses , FUN = function(ll){ ll$se^2 } )
# Rubin inference
tau_comb <- mitools::MIcombine( betas , vars )
summary(tau_comb)

## End(Not run)
```

`micombine.F` *Combination of F Statistics for Multiply Imputed Datasets Using a Chi Square Approximation*

Description

Several F statistics from multiply imputed datasets are combined using an approximation based on χ^2 statistics (see `micombine.chisquare`).

Usage

```
micombine.F(Fvalues, df1, display = TRUE)
```

Arguments

`Fvalues`	Vector containing F values.
`df1`	Degrees of freedom of the denominator. Degrees of freedom of the numerator are approximated by ∞ (large number of degrees of freedom).
`display`	A logical indicating whether results should be displayed at the console

Value

The same output as in `micombine.chisquare`

Author(s)

Alexander Robitzsch

See Also

`micombine.chisquare`

Examples

```
##############################################################################
# EXAMPLE 1: F statistics for 5 imputed datasets
##############################################################################

Fvalues <- c( 6.76 , 4.54 , 4.23 , 5.45 , 4.78 )
micombine.F(Fvalues, df1=4 )
  ##   Combination of Chi Square Statistics for Multiply Imputed Data
  ##   Using 5 Imputed Data Sets
  ##   F(4,67.11)=4.097     p=0.00497
  ##   Chi Square Approximation Chi2(4)=16.387     p=0.00254
```

`mids2datlist` *Converting a* `mids`*,* `mids.1chain` *or* `mids.nmi` *Object in a Dataset List*

Description

Converts a `mids` `mids.1chain` or `mids.nmi` object in a dataset list.

Usage

```
mids2datlist(midsobj)
```

Arguments

`midsobj`	Object of class `mids`, `mids.1chain` or `mids.nmi`

Value

List of multiply imputed datasets

Author(s)

Alexander Robitzsch

Examples

```
## Not run:
##############################################################################
# EXAMPLE 1: Imputing nhanes data and convert result into a dataset list
##############################################################################

data(nhanes,package="mice")

#**** imputation using mice
imp1 <- mice( nhanes , m=3 , maxit=5 )
# convert mids object into list
datlist1 <- mids2datlist( imp1 )
```

```
#**** imputation using mice.1chain
imp2 <- mice.1chain( nhanes , burnin=4 , iter=20 , Nimp=5 )
# convert mids.1chain object into list
datlist2 <- mids2datlist( imp2 )

################################################################################
# EXAMPLE 2: Nested multiple imputation and datalist conversion
################################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2
   # list of 5 datasets containing 5 plausible values

# remove first four variables
M <- length(datlist)
for (ll in 1:M){
    datlist[[ll]] <- datlist[[ll]][ , -c(1:4) ]
                }

#***************
# (1) nested multiple imputation using mice
imp1 <- mice.nmi( datlist ,  m=4 , maxit=3 )
summary(imp1)

#***************
# (2) nested multiple imputation using mice.1chain
imp2 <- mice.nmi( datlist , Nimp=4 , burnin=10 ,iter=22, type="mice.1chain")
summary(imp2)

#**************
# conversion into a datalist
datlist.i1 <- mids2datlist( imp1 )
datlist.i2 <- mids2datlist( imp2 )

## End(Not run)
```

`mids2mlwin` *Export* `mids` *object to MLwiN*

Description

Converts a `mids` object into a format recognized by the multilevel software MLwiN.

Usage

```
mids2mlwin(imp, file.prefix, path = getwd(), sep = " ", dec = ".", silent = FALSE)
```

Arguments

imp	The imp argument is an object of class mids, typically produced by the mice() function.
file.prefix	A character string describing the prefix of the output data files.
path	A character string containing the path of the output file. By default, files are written to the current R working directory.
sep	The separator between the data fields.
dec	The decimal separator for numerical data.
silent	A logical flag stating whether the names of the files should be printed.

Value

The return value is NULL.

Author(s)

Thorsten Luka <mail@thorstenluka.de>

Examples

```
## Not run:
# imputation nhanes data
data(nhanes)
imp <- mice(nhanes)
# write files to MLwiN
mids2mlwin(imp, file.prefix="nhanes" )

## End(Not run)
```

NestedImputationList *Functions for Analysis of Nested Multiply Imputed Datasets*

Description

The function NestedImputationList takes a list of lists of datasets and converts this into an object of class NestedImputationList.

Statistical models can be estimated with the function with.NestedImputationList.

The mitools::MIcombine method can be used for objects of class NestedImputationResultList which are the output of with.NestedImputationList.

Usage

```
NestedImputationList( datasets )

## S3 method for class 'NestedImputationResultList'
MIcombine(results, ...)
```

Arguments

datasets	List of lists of datasets which are created by nested multiple imputation.
results	Object of class NestedImputationResultsList
...	Further arguments to be passed.

Value

Function NestedImputationList: Object of class NestedImputationList.

Function MIcombine.NestedImputationList: Object of class mipo.nmi.

Author(s)

Alexander Robitzsch

See Also

with.NestedImputationList, within.NestedImputationList, pool.mids.nmi, NMIcombine

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Nested multiple imputation and conversion into an object of class
#            NestedImputationList
#############################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2

# remove first four variables
M <- length(datlist)
for (ll in 1:M){
    datlist[[ll]] <- datlist[[ll]][ , -c(1:4) ]
                }

# nested multiple imputation using mice
imp1 <- mice.nmi( datlist ,  m=3 , maxit=2 )
summary(imp1)

# create object of class NestedImputationList
datlist1 <- mids2datlist( imp1 )
datlist1 <- NestedImputationList( datlist1 )

# estimate linear model using with
res1 <- with( datlist1 , lm( ASMMAT ~ female + migrant ) )
# pool results
mres1 <- MIcombine( res1 )
summary(mres1)
coef(mres1)
```

```
vcov(mres1)

## End(Not run)
```

NMIwaldtest *Wald Test for Nested Multiply Imputed Datasets*

Description

Performs a Wald test for nested multiply imputed datasets (NMIwaldtest) and ordinary multiply imputed datasets (MIwaldtest), see Reiter and Raghanuthan (2007). The correspondent statistic is also called the D_1 statistic.

The function create.designMatrices.waldtest is a helper function for the creation of design matrices.

Usage

```
NMIwaldtest(qhat, u, Cdes = NULL, rdes = NULL, testnull = NULL)

MIwaldtest(qhat, u, Cdes = NULL, rdes = NULL, testnull = NULL)

## S3 method for class 'NMIwaldtest'
summary(object,digits=4,...)

## S3 method for class 'MIwaldtest'
summary(object,digits=4,...)

create.designMatrices.waldtest( pars , k )
```

Arguments

qhat	List or array of estimated parameters
u	List or array of estimated covariance matrices of parameters
Cdes	Design matrix C for parameter test (see Details)
rdes	Constant vector r (see Details)
testnull	Vector containing names of parameters which should be tested for a parameter value of zero.
object	Object of class NMIwaldtest
digits	Number of digits after decimal for print
...	Further arguments to be passed
pars	Vector of parameter names
k	Number of linear hypotheses which should be tested

Details

The Wald test is performed for a linear hypothesis $C\boldsymbol{\theta} = r$ for a parameter vector $\boldsymbol{\theta}$.

Value

List with following entries

`stat`	Data frame with test statistic
`qhat`	Transformed parameter according to linear hypothesis
`u`	Covariance matrix of transformed parameters

Note

The function `create.designMatrices.waldtest` is a helper function for the creation of design matrices.

Author(s)

Alexander Robitzsch

References

Reiter, J. P. and Raghunathan, T. E. (2007). The multiple adaptations of multiple imputation. *Journal of the American Statistical Association*, **102**, 1462-1471.

See Also

NMIcombine

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Nested multiple imputation and Wald test | TIMSS data
#############################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2
# remove first four variables
M <- length(datlist)
for (ll in 1:M){
    datlist[[ll]] <- datlist[[ll]][ , -c(1:4) ]
                }

#***************
# (1) nested multiple imputation using mice
imp1 <- mice.nmi( datlist ,  m=3 , maxit=2 )
summary(imp1)

#**** Model 1: Linear regression with interaction effects
```

```
res1 <- with( imp1 , lm( likesc ~ female*migrant + female*books  ) )
pres1 <- pool.mids.nmi( res1 )
summary(pres1)

# test whether both interaction effects equals zero
pars <- dimnames(pres1$qhat)[[3]]
des <- create.designMatrices.waldtest( pars = pars , k=2)
Cdes <- des$Cdes
rdes <- des$rdes
Cdes[1, "female:migrant"] <- 1
Cdes[2, "female:books"] <- 1
wres1 <- NMIwaldtest( qhat=pres1$qhat , u=pres1$u , Cdes=Cdes , rdes=rdes )
summary(wres1)

# a simpler specification is the use of "testnull"
testnull <- c("female:migrant" , "female:books")
wres1b <- NMIwaldtest( qhat=qhat , u=u , testnull=testnull )
summary(wres1b)

#**** Model 2: Multivariate linear regression
res2 <- with( imp1 , lm( cbind( ASMMAT , ASSSCI ) ~
                          0 + I(1*(female==1)) + I(1*(female==0))   ) )
pres2 <- pool.mids.nmi( res2 )
summary(pres2)

# test whether both gender differences equals -10 points
pars <- dimnames(pres2$qhat)[[3]]
  ##  > pars
  ##  [1] "ASMMAT:I(1 * (female == 1))" "ASMMAT:I(1 * (female == 0))"
  ##  [3] "ASSSCI:I(1 * (female == 1))" "ASSSCI:I(1 * (female == 0))"

des <- create.designMatrices.waldtest( pars = pars , k=2)
Cdes <- des$Cdes
rdes <- c(-10,-10)
Cdes[1, "ASMMAT:I(1*(female == 1))"] <- 1
Cdes[1, "ASMMAT:I(1*(female == 0))"] <- -1
Cdes[2, "ASSSCI:I(1*(female == 1))"] <- 1
Cdes[2, "ASSSCI:I(1*(female == 0))"] <- -1

wres2 <- NMIwaldtest( qhat=pres2$qhat , u=pres2$u , Cdes=Cdes , rdes=rdes )
summary(wres2)

# test only first hypothesis
wres2b <- NMIwaldtest( qhat=pres2$qhat , u=pres2$u , Cdes=Cdes[1,,drop=FALSE] ,
                        rdes=rdes[1] )
summary(wres2b)

#############################################################################
# EXAMPLE 2: Multiple imputation and Wald test | TIMSS data
#############################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
```

```
dat <- data.timss2[[1]]
dat <- dat[ , - c(1:4) ]

# perform multiple imputation
imp <- mice::mice( dat , m=6 , maxit=3 )

# define analysis model
res1 <- with( imp , lm( likesc ~ female*migrant + female*books  ) )
pres1 <- mice::pool( res1 )
summary(pres1)

# Wald test for zero interaction effects
qhat <- pres1$qhat
u <- pres1$u
pars <- dimnames(pres1$qhat)[[2]]
des <- create.designMatrices.waldtest( pars = pars , k=2)
Cdes <- des$Cdes
rdes <- des$rdes
Cdes[1, "female:migrant"] <- 1
Cdes[2, "female:books"] <- 1

# apply MIwaldtest function
wres1 <- MIwaldtest( qhat , u , Cdes , rdes )
summary(wres1)

# use again "testnull"
testnull <- c("female:migrant" , "female:books")
wres1b <- MIwaldtest( qhat=qhat , u=u , testnull=testnull )
summary(wres1b)

#***** linear regression with cluster robust standard errors

# convert object of class mids into a list object
datlist_imp <- mids2datlist( imp )
# define cluster
idschool <- as.numeric( substring( data.timss2[[1]]$IDSTUD , 1 , 5 ) )
# linear regression
res2 <- lapply( datlist_imp , FUN = function(data){
           lm.cluster( data=data , formula=likesc ~ female*migrant + female*books ,
                           cluster= idschool ) } )
# extract parameters and covariance matrix
qhat <- lapply( res2 , FUN = function(rr){ coef(rr) } )
u <- lapply( res2 , FUN = function(rr){ vcov(rr) } )
# perform Wald test
wres2 <- MIwaldtest( qhat , u , Cdes , rdes )
summary(wres2)

## End(Not run)
```

Description

This function does some formatting of output.

Usage

```
output.format1(stringtype , label , rep.N=1 ,stringlength = 70)
```

Arguments

stringtype	Type of string for display, e.g. "*", "-", ...
label	Some comment which should be displayed at the console
rep.N	Number of lines which shall be left blank
stringlength	Length of vector with label

Value

Generates a string output at the R console

Author(s)

Alexander Robitzsch

Examples

```
output.format1(stringtype="*'" , label="HELLO WORLD" , stringlength = 20)
##   *'*'*'*'*'*'*'*'*'*'*'*'*'*'*'*'*'*'*'*'
##   HELLO WORLD
```

---

pca.covridge *Principal Component Analysis with Ridge Regularization*

Description

Performs a principal component analysis for a dataset while a ridge parameter is added on the diagonal of the covariance matrix.

Usage

```
pca.covridge(x, ridge = 10^(-10))
```

Arguments

x	A numeric matrix
ridge	Ridge regularization parameter for the covariance matrix

Value

A list with following entries:

`loadings`	Matrix of factor loadings
`scores`	Matrix of principal component scores
`sdev`	Vector of standard deviations of factors (square root of eigenvalues)

Author(s)

Alexander Robitzsch

See Also

Principal component analysis in **stats**: `stats::princomp`

For calculating first eigenvalues of a symmetric matrix see also `sirt::eigenvalues.sirt` in the **sirt** package.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: PCA on imputed internet data
#############################################################################

library(mice)
data(data.internet)
dat <- as.matrix( data.internet)

# single imputation in mice
imp <- mice( dat , m=1 , maxit=10 )

# apply PCA
pca.imp <- pca.covridge( complete(imp) )
  ##   > pca.imp$sdev
  ##      Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7
  ##   3.0370905 2.3950176 2.2106816 2.0661971 1.8252900 1.7009921 1.6379599

# compare results with princomp
pca2.imp <- princomp( complete(imp) )
  ##   > pca2.imp
  ##   Call:
  ##   princomp(x = complete(imp))
  ##
  ##   Standard deviations:
  ##      Comp.1    Comp.2    Comp.3    Comp.4    Comp.5    Comp.6    Comp.7
  ##   3.0316816 2.3907523 2.2067445 2.0625173 1.8220392 1.6979627 1.6350428

## End(Not run)
```

pool.mids.nmi *Pooling for Nested Multiple Imputation*

Description

Statistical inference for scalar parameters for nested multiply imputed datasets (Rubin, 2003; Harel & Schafer, 2003; Reiter & Raghanuthan, 2007).

The NMIcombine and NMIextract functions are extensions of mitools::MIcombine and mitools::MIextract.

Usage

```
pool.mids.nmi(object, method = "largesample")

NMIcombine( qhat , u=NULL , NMI=TRUE, comp_cov=TRUE, is_list=TRUE )

NMIextract(results, expr, fun)

## S3 method for class 'mipo.nmi'
summary(object, digits=3, ...)

## S3 method for class 'mipo.nmi'
coef(object, ...)

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

Arguments

object	Object of class mids.nmi. For summary it must be an object of class mipo.nmi.
method	Method for calculating degrees of freedom. Until now, only the method "largesample" is available.
qhat	List of lists of parameter estimates. In case of an ordinary imputation it can only be a list.
u	Optional list of lists of covariance matrices of parameter estimates
NMI	Optional logical indicating whether the NMIcombine function should be applied for results of nested multiply imputed datasets. It is set to FALSE if only a list results of multiply imputed datasets is available.
comp_cov	Optional logical indicating whether covariances between parameter estimates should be estimated.
is_list	Optional logical indicating whether qhat and u are provided as lists as an input. If is_list=FALSE, appropriate arrays can be used as inpur.
results	A list of objects
expr	An expression
fun	A function of one argument

digits	Number of digits after decimal for printing results in summary.
...	Further arguments to be passed.

Value

Object of class mipo.nmi with following entries

qhat	Estimated parameters in all imputed datasets
u	Estimated covariance matrices of parameters in all imputed datasets
qbar	Estimated parameter
ubar	Average estimated variance within imputations
Tm	Total variance of parameters
df	Degrees of freedom
lambda	Total fraction of missing information
lambda_Between	Fraction of missing information of between imputed datasets (first stage imputation)
lambda_Within	Fraction of missing information of within imputed datasets (second stage imputation)

Author(s)

Alexander Robitzsch

References

Harel, O., & Schafer, J. (2003). *Multiple imputation in two stages*. In Proceedings of Federal Committee on Statistical Methodology 2003 Conference.

Reiter, J. P., & Raghunathan, T. E. (2007). The multiple adaptations of multiple imputation. *Journal of the American Statistical Association*, **102(480)**, 1462-1471.

Rubin, D. B. (2003). Nested multiple imputation of NMES via partially incompatible MCMC. *Statistica Neerlandica*, **57(1)**, 3-18.

See Also

mice.nmi

mice::pool, mitools::MIcombine, mitools::MIextract

MIcombine.NestedImputationResultList

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Nested multiple imputation and statistical inference
#############################################################################

library(BIFIEsurvey)
```

```
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2
# remove first four variables
M <- length(datlist)
for (ll in 1:M){
    datlist[[ll]] <- datlist[[ll]][ , -c(1:4) ]
                }

#***************
# (1) nested multiple imputation using mice
imp1 <- mice.nmi( datlist ,  m=3 , maxit=2 )
summary(imp1)

#***************
# (2) first linear regression: ASMMAT ~ migrant + female
res1 <- with( imp1 , lm( ASMMAT ~ migrant + female ) ) # fit
pres1 <- pool.mids.nmi( res1 )  # pooling
summary(pres1)  # summary
coef(pres1)
vcov(pres1)

#***************
# (3) second linear regression: likesc ~ migrant + books
res2 <- with( imp1 , lm( likesc ~ migrant + books  ) )
pres2 <- pool.mids.nmi( res2 )
summary(pres2)

#***************
# (4) some descriptive statistics using the mids.nmi object
res3 <- with( imp1 , c( "M_lsc"=mean(likesc) , "SD_lsc"=sd(likesc) ) )
pres3 <- NMIcombine( qhat = res3$analyses )
summary(pres3)

#*************
# (5) apply linear regression based on imputation list

# convert mids object to datlist
datlist2 <- mids2datlist( imp1 )
str(datlist2, max.level=1)

# double application of lapply to the list of list of nested imputed datasets
res4 <- lapply( datlist2 , FUN = function(dl){
    lapply( dl , FUN = function(data){
            lm( ASMMAT ~ migrant + books  , data = data )
                            } )
                }  )

# extract coefficients
qhat <- lapply( res4 , FUN = function(bb){
            lapply( bb , FUN = function(ww){
                    coef(ww)
                        } )
                } )
```

```
# shorter function
NMIextract( results=res4 , fun=coef )

# extract covariance matrices
u <- lapply( res4 , FUN = function(bb){
            lapply( bb , FUN = function(ww){
                    vcov(ww)
                        } )
                } )
# shorter function
NMIextract( results=res4 , fun=vcov )

# apply statistical inference using the NMIcombine function
pres4 <- NMIcombine( qhat=qhat , u=u )
summary(pres4)

## End(Not run)
```

Reval R *Utilities: Evaluates a String as an Expression in* R

Description

This function evaluates a string as an R expression.

Usage

```
Reval(Rstring, print.string=TRUE)

Revalpr(Rstring, print.string=TRUE) # = Reval( print(Rstring) )

Revalprstr(Rstring, print.string=TRUE)   #  = Reval( print(str(Rstring)) )
```

Arguments

Rstring	String which shall be evaluated in R
print.string	Should the string printed on the console?

Details

The string is evaluated in the parent environment. See base::eval for the definition of environments in R.

Author(s)

Alexander Robitzsch

Examples

```
# This function is simply a shortage function
# See the definition of this function:
Reval <- function( Rstring , print.string=TRUE){
    if (print.string){ cat( paste( Rstring ) , "\n"  ) }
     eval.parent( parse( text = paste( Rstring )) , n=1 )
            }

Reval( "a <- 2^3" )
  ## a <- 2^3
a
  ## [1] 8
```

`Rhat.mice` *Rhat Convergence Statistic of a* `mice` *Imputation*

Description

Computes the Rhat statistic for a `mids` object.

Usage

```
Rhat.mice(mice.object)
```

Arguments

`mice.object`	Object of class `mids`

Value

Data frame containing the Rhat statistic for mean and variances for all variables of the Markov chains used for imputation

Author(s)

Alexander Robitzsch

References

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Rhat statistic for nhanes data
#############################################################################

library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes 3 parallel chains
imp1 <- mice( nhanes ,m=3 , maxit=10 , imputationMethod=rep("norm" , 4 ) )
Rhat.mice( imp1 )
  ##     variable MissProp Rhat.M.imp Rhat.Var.imp
  ##   1      bmi       36  1.0181998     1.155807
  ##   2      hyp       32  1.0717677     1.061174
  ##   3      chl       40  0.9717109     1.318721

## End(Not run)
```

---

round2 R *Utilities: Rounding DIN 1333 (Kaufmaennisches Runden)*

Description

This is a rounding function which rounds up for all numbers according to the rule of 'kaufmaennisches Runden' (DIN 1333).

Usage

```
round2(vec, digits = 0)
```

Arguments

vec	Numeric vector
digits	Number of digits after decimal fort rounding

Value

Vector with rounded values

Author(s)

Alexander Robitzsch

Examples

```
#*** EXAMPLE 1
vec <- c( 1.5 , 2.5 , 3.5 , 1.51 ,  1.49)
vec
round(vec)
round2(vec)
  ##   > vec
  ##   [1] 1.50 2.50 3.50 1.51 1.49
  ##   > round(vec)
  ##   [1] 2 2 4 2 1
  ##   > round2(vec)
  ##   [1] 2 3 4 2 1

#*** EXAMPLE 2
vec <- - c( 1.5 , 2.5 , 3.5 , 1.51 ,  1.49)
vec
round(vec)
round2(vec)
  ##   > vec
  ##   [1] -1.50 -2.50 -3.50 -1.51 -1.49
  ##   > round(vec)
  ##   [1] -2 -2 -4 -2 -1
  ##   > round2(vec)
  ##   [1] -2 -3 -4 -2 -1

#*** EXAMPLE 3
vec <- c(8.4999999 , 8.5 , 8.501 , 7.4999999 , 7.5 , 7.501 )
round(vec)
round2( vec )
round2( vec , digits=1)
round2( -vec )
  ##   > round(vec)
  ##   [1] 8 8 9 7 8 8
  ##   > round2( vec )
  ##   [1] 8 9 9 7 8 8
  ##   > round2( vec , digits=1)
  ##   [1] 8.5 8.5 8.5 7.5 7.5 7.5
  ##   > round2( -vec )
  ##   [1] -8 -9 -9 -7 -8 -8
```

Rsessinfo R *Utilities:* R *Session Information*

Description

Informs about current R session.

Usage

```
Rsessinfo()
```

Value

A string containing reduced information about R session info

Author(s)

Alexander Robitzsch

Examples

```
Rsessinfo()
  ##   > Rsessinfo()
  ##   [1] "R version 2.15.2 (2012-10-26) x86_64, mingw32 | nodename = SD70 | login = robitzsch"
```

---

save.data R *Utilities: Saving/Writing Data Files using* **miceadds**

Description

This function is a wrapper function for saving or writing data frames or matrices.

Usage

```
save.data( data, filename, type="Rdata", path=getwd(), row.names=FALSE, na=NULL,
      suffix = NULL , suffix_space = "__" , ...)
```

Arguments

data	Data frame or matrix to be saved
filename	Name of data file
type	The type of file in which the data frame or matrix should be loaded. This can be Rdata (for R binary format, using base::save, csv (using utils::write.csv2), csv (using utils::write.csv), table (using utils::write.table) or sav (using sjmisc::write_spss).
path	Directory from which the dataset should be loaded
row.names	Optional logical indicating whether row names should be included in saved csv or csv2 files.
na	Missing value handling. The default is "" for type="csv" and type="csv2" and is "." for type="table".
suffix	Optional suffix in file name.
suffix_space	Optional place holder if a suffix is used.
...	Further arguments to be passed to save, write.csv2, write.csv, write.table or sjmisc::write_spss.

Author(s)

Alexander Robitzsch

See Also

See load.Rdata and load.data for saving/writing R data frames.

Examples

```
## Not run:
#*** use data.ma01 as an example for writing data files using save.data
data(data.ma01)
dat <- data.ma01

# set a working directory
pf2 <- "P:/ARb/temp_miceadds"

# save data in Rdata format
save.data( dat , filename="ma01data"  , type="Rdata" , path=pf2)

# save data in table format
save.data( dat , filename="ma01data"  , type="table" , path=pf2 ,
            row.names=FALSE , na =".")

# save data in csv2 format
save.data( dat , filename="ma01data"  , type="csv2" , path=pf2 ,
            row.names=FALSE , na ="")

# save data in sav format
save.data( dat , filename="ma02data"  , type="sav" ,  path=pf2 )

## End(Not run)
```

---

save.Rdata R *Utilities: Save a Data Frame in* Rdata *Format*

Description

This function saves a data frame in a Rdata format.

Usage

```
save.Rdata(dat, name, path = NULL, part.numb = 1000)
```

Arguments

dat	Data frame
name	Name of the R object to be saved
path	Directory for saving the object
part.numb	Number of rows of the data frame which should also be saved in csv format. The default is saving 1000 rows.

Author(s)

Alexander Robitzsch

Examples

```
## Not run:
dfr <- matrix( 2*1:12-3 , 4 ,3 )
save.Rdata( dfr , "dataframe_test" )
## End(Not run)
```

---

scan.vec R *Utilities: Scan a Character Vector*

Description

This function splits a string into a character vector.

Usage

```
scan.vec(vec)
scan.vector(vec)
```

Arguments

vec	A string which should be splitted according to blanks

Author(s)

Alexander Robitzsch

Examples

```
vars <- scan.vector( "female urbgrad  groesse  Nausg    grpgroesse   privat  ")
  ## Read 6 items
vars
  ## [1] "female"     "urbgrad"    "groesse"    "Nausg"      "grpgroesse"
  ## [6] "privat"
```

source.all R *Utilities: Source All R Files Within a Directory*

Description

Sources all R files within a specified directory.

Usage

```
source.all( path , grepstring="\\.R",  print.source=TRUE )
```

Arguments

path	Path where the files are located
grepstring	Which strings should be looked for? grepstring can also be a vector.
print.source	An optional logical whether the source process printed on the console?

Author(s)

Alexander Robitzsch

Examples

```
## Not run:
# define path
path <- "c:/myfiles/"
# source all files containing the string 'Rex'
source.all( path , "Rex" )
## End(Not run)
```

str_C.expand.grid R *Utilities: String Paste Combined with* expand.grid

Description

String paste combined with expand.grid

Usage

```
str_C.expand.grid(xlist , indices=NULL)
```

Arguments

xlist	A list of character vectors
indices	Optional vector of indices to be permuted in xlist

Value

A character vector

Author(s)

Alexander Robitzsch

Examples

```
#############################################################################
# EXAMPLE 1: Some toy examples
#############################################################################

x1 <- list( c("a","b" ) , c("t", "r","v") )
str_C.expand.grid( x1 )
  ##   [1] "at" "bt" "ar" "br" "av" "bv"

x1 <- list( c("a","b" ) , paste0("_" , 1:4 ) , c("t", "r","v") )
str_C.expand.grid( x1 , indices=c(2,1,3) )
  ##    [1] "_1at" "_1bt" "_2at" "_2bt" "_3at" "_3bt" "_4at" "_4bt" "_1ar" "_1br"
  ##   [11] "_2ar" "_2br" "_3ar" "_3br" "_4ar" "_4br" "_1av" "_1bv" "_2av" "_2bv"
  ##   [21] "_3av" "_3bv" "_4av" "_4bv"

## Not run:
##**************************************************************************
## The function 'str_C.expand.grid' is currently defined as
function( xlist , indices=NULL ){
     xeg <- expand.grid( xlist)
     if ( ! is.null(indices) ){ xeg <- xeg[ , indices ]}
     apply( xeg , 1 , FUN = function(vv){ paste0( vv , collapse="") } )
}
##**************************************************************************

## End(Not run)
```

`sumpreserving.rounding` *Sum Preserving Rounding*

Description

This function implements sum preserving rounding. If the supplied data is a matrix, then the sum of all row entries is preserved.

Usage

```
sumpreserving.rounding(data, digits=0, preserve=TRUE)
```

Arguments

data	Vector or data frame
digits	Number of digits to be round
preserve	Should the sum be preserved?

Author(s)

Alexander Robitzsch

Examples

```
###################################
# Example 1
###################################

# define example data
data <- c( 1455 , 1261 , 1067 , 970 , 582 , 97 )
data <- 100 * data / sum(data)

( x1 <- round( data ) )
sum(x1)
(x2 <- sumpreserving.rounding( data ) )
sum(x2)

  ##   > ( x1 <- round( data ) )
  ##   [1] 27 23 20 18 11  2
  ##   > sum(x1)
  ##   [1] 101
  ##   > (x2 <- sumpreserving.rounding( data ) )
  ##   [1] 27 23 20 18 10  2
  ##   > sum(x2)
  ##   [1] 100

###################################
# Example 2
###################################

# matrix input
data <- rbind( data , data )
( x1 <- round( data ) )
rowSums(x1)
(x2 <- sumpreserving.rounding( data ) )
rowSums(x2)

###################################
# Example 3
###################################

x2 <- c( 1.4 , 1.4 , 1.2 )
round(x2)
sumpreserving.rounding(x2)
```

```
## > round(x2)
## [1] 1 1 1
## > sumpreserving.rounding(x2)
## [1] 1 2 1
```

`systime` R *Utilities: Various Strings Representing System Time*

Description

This function generates system time strings in several formats.

Usage

```
systime()
```

Value

A vector with entries of system time (see Examples).

Author(s)

Alexander Robitzsch

Examples

```
systime()
  ## > systime()
  ## [1] "2015-05-14 17:38:02" "2015-05-14"          "20150514"
  ## [4] "2015-05-14_1738"     "2015-05-14_1700"     "20150514_173802"
  ## [7] "20150514173802"      "SD70_20150514173802"
```

`tw.imputation` *Two-Way Imputation*

Description

Two-way imputation using the simple method of Sijtsma and van der Ark (2003) and the MCMC based imputation of van Ginkel et al. (2007).

Usage

```
tw.imputation(data, integer = FALSE)

tw.mcmc.imputation(data, iter = 100, integer = FALSE)
```

Arguments

data	Matrix of item responses corresponding to a scale
integer	A logical indicating whether imputed values should be integers. The default is FALSE.
iter	Number of iterations

Details

For persons p and items i, the two-way imputation is conducted by posing a linear model of tau-equivalent measurements:

$$X_{pi} = \theta_p + b_i + \varepsilon_{ij}$$

If the score X_{pi} is missing then it is imputed by

$$\hat{X}_{pi} = \tilde{X}_p + b_i$$

where $\tilde{X}_p$ is the person mean of person p of the remaining items with observed responses.

The two-way imputation can also be seen as a scaling procedure to obtain a scale score which takes different item means into account.

Value

A matrix with original and imputed values

Author(s)

Alexander Robitzsch

References

Sijtsma, K., & Van der Ark, L. A. (2003). Investigation and treatment of missing item scores in test and questionnaire data. *Multivariate Behavioral Research*, **38**, 505-528.

Van Ginkel, J. R., Van der Ark, A., Sijtsma, K., & Vermunt, J. K. (2007). Two-way imputation: A Bayesian method for estimating missing scores in tests and questionnaires, and an accurate approximation. *Computational Statistics & Data Analysis*, **51**, 4013-4027.

Examples

```
############################################################################
# EXAMPLE 1: Two-way imputation data.internet
############################################################################

data(data.internet)
data <- data.internet

#***
# Model 1: Two-way imputation method of Sijtsma and van der Ark (2003)
set.seed(765)
dat.imp <- tw.imputation( data )
dat.imp[ 278:281,]
```

```
  ##       IN9     IN10    IN11     IN12
  ##   278   5 4.829006 5.00000 4.941611
  ##   279   5 4.000000 4.78979 4.000000
  ##   280   7 4.000000 7.00000 7.000000
  ##   281   4 3.000000 5.00000 5.000000

## Not run:
#***
# Model 2: Two-way imputation method using MCMC
dat.imp <- tw.mcmc.imputation( data , iter=3)
dat.imp[ 278:281,]
  ##       IN9     IN10     IN11     IN12
  ##   278   5 6.089222 5.000000 3.017244
  ##   279   5 4.000000 5.063547 4.000000
  ##   280   7 4.000000 7.000000 7.000000
  ##   281   4 3.000000 5.000000 5.000000

## End(Not run)
```

`visitSequence.determine`

Automatic Determination of a Visit Sequence in `mice`

Description

This function automatically determines a visit sequence for a specified model in `mice::mice` when passive variables are defined as imputation methods. Note that redundant visits could be computed and a user should check the plausibility of the result.

Usage

```
visitSequence.determine(impMethod, vis, data, maxit = 10)
```

Arguments

`impMethod`	Vector with imputation methods
`vis`	Initial vector of visit sequence
`data`	Data frame to be used for multiple imputations
`maxit`	Maximum number of iteration for computation of the updated visit sequence

Value

Updated vector of the visit sequence

Author(s)

Alexander Robitzsch

See Also

mice::mice

Examples

```
#############################################################################
# EXAMPLE 1: Visit sequence for a small imputation model
#############################################################################

data( data.smallscale )
# select a small number of variables
dat <- data.smallscale[ , paste0("v",1:4) ]
V <- ncol(dat)

# define initial vector of imputation methods
impMethod <- rep("norm" , V)
names(impMethod) <- colnames(dat)
# define variable names and imputation method for passive variables in a data frame
dfr.impMeth <- data.frame( "variable" = NA ,
                  "impMethod" =  NA )
dfr.impMeth[1,] <- c("v1_v1" , "~ I(v1^2)" )
dfr.impMeth[2,] <- c("v2_v4" , "~ I(v2*v4)" )
dfr.impMeth[3,] <- c("v4log" , "~ I( log(abs(v4)))" )
dfr.impMeth[4,] <- c("v12" , "~ I( v1 + v2 + 3*v1_v1 - v2_v4 )" )
# add variables to dataset and imputation methods
VV <- nrow(dfr.impMeth)
for (vv in 1:VV){
    impMethod[ dfr.impMeth[vv,1] ] <- dfr.impMeth[vv,2]
    dat[ , dfr.impMeth[vv,1] ] <- NA
                }

# run empty imputation model to obtain initial vector of visit sequence
imp0 <- mice::mice( dat , m=1 , imputationMethod = impMethod , maxit=0 )
imp0$vis

# update visit sequence
vis1 <- visitSequence.determine( impMethod=impMethod , vis=imp0$vis , data = dat)

# imputation with updated visit sequence
imp <- mice::mice( dat, m=1, imputationMethod=impMethod, visitSequence=vis1, maxit=2)
```

---

with.miceadds *Evaluates an Expression for (Nested) Multiply Imputed Datasets*

Description

Evaluates an expression for (nested) multiply imputed datasets. These functions extend the following functions: mice::with.mids, base::within.data.frame, mitools::with.imputationList.

Usage

```
## S3 method for class 'mids.1chain'
with(data, expr, ...)

## S3 method for class 'mids.nmi'
with(data, expr, ...)

## S3 method for class 'imputationList'
within(data, expr, ...)

## S3 method for class 'NestedImputationList'
with(data, expr, fun , ...)

## S3 method for class 'NestedImputationList'
within(data, expr, ...)

## S3 method for class 'mira.nmi'
summary(object , ...)
```

Arguments

data	Object of class mids.1chain, mids.nmi, imputationList or NestedImputationList
expr	Expression with a formula object.
fun	A function taking a data frame argument
...	Additional parameters to be passed to expr.
object	Object of class mira.nmi.

Value

with.mids.1chain: List of class mira.

with.mids.nmi: List of class mira.nmi.

within.imputationList: List of class imputationList.

with.NestedImputationList: List of class NestedImputationResultList.

within.NestedImputationList: List of class NestedImputationList.

Author(s)

Slightly modified code of mice::with.mids, mice::summary.mira, base::within.data.frame

See Also

mice::with.mids, mice::summary.mira, base::within.data.frame, mitools::with.imputationList

Imputation functions in **miceadds**: mice.1chain, mice.nmi

Examples

```
#############################################################################
# EXAMPLE 1: One chain nhanes data | application of 'with' and 'within'
#############################################################################

library(mice)
data(nhanes, package="mice")
set.seed(9090)

# nhanes data in one chain
imp <- mice.1chain( nhanes , burnin=5 , iter=40 , Nimp=4 )
# apply linear regression
res <- with( imp , expr = lm( hyp ~ age + bmi  ) )
summary(res)
# pool results
summary(pool(res))

# calculate some descriptive statistics
res2 <- with( imp , expr = c( mean(hyp) , sd(age) ) )
res2

# convert mids object into an object of class imputationList
datlist <- mids2datlist( imp )
datlist <- mitools::imputationList(datlist)

# define formulas for modification of the data frames in imputationList object
datlist2 <- within.imputationList( datlist , {
                     age.D3 <- 1*(age == 3)
                     hyp_chl <- hyp * chl
                        } )
# look at modified dataset
head( datlist2$imputations[[1]] )

## Not run:
#############################################################################
# EXAMPLE 2: Nested multiple imputation and application of with/within methods
#############################################################################

library(BIFIEsurvey)
data(data.timss2 , package="BIFIEsurvey" )
datlist <- data.timss2

# remove first four variables
M <- length(datlist)
for (ll in 1:M){
    datlist[[ll]] <- datlist[[ll]][ , -c(1:4) ]
                }

# nested multiple imputation using mice
imp1 <- mice.nmi( datlist ,  m=4 , maxit=3 )
summary(imp1)
```

```
# apply linear model and use summary method for all analyses of imputed datasets
res1 <- with( imp1 , lm( ASMMAT ~ migrant + female ) )
summary(res1)

# convert mids.nmi object into an object of class NestedImputationList
datlist1 <- mids2datlist( imp1 )
datlist1 <- NestedImputationList( datlist1 )

# use with function
res1b <- with( datlist1 , glm( ASMMAT ~ migrant + female ) )

# use within function for data transformations
datlist2 <- within( datlist1 , {
            highsc <- 1*(ASSSCI > 600)
            books_dum <- 1*(books>=3)
            rm(scsci)   # remove variable scsci
                    } )

## End(Not run)
```

`write.mice.imputation` *Export Multiply Imputed Datasets from a* `mids` *Object*

Description

Exports multiply imputed datasets and information about the imputation. Objects of class `mids` (generated by `mice::mice`) and `mids.1chain` (generated by `mice.1chain`) are supported.

Usage

```
write.mice.imputation(mi.res, name, include.varnames = TRUE,
      long = TRUE, mids2spss = TRUE, spss.dec = ",", dattype = NULL)
```

Arguments

`mi.res`	Object of class `mids` or `mids.1chain`
`name`	Name of created folder and datasets
`include.varnames`	An optional logical indicating whether variable names should be included in the imputed dataset. The default is `TRUE`.
`long`	An optional logical indicating whether the dataset should also be saved in a long format?
`mids2spss`	An optional logical indicating whether a syntax for reading imputed datasets in SPSS should be included
`spss.dec`	SPSS decimal separator (can be `","` or `"."`)
`dattype`	Format of the saved dataset: `csv` or `csv2`

Value

Several files are saved using `impxxx` (the `name`) as the prefix:

`impxxx.Rdata` Saved object of class `mids`

`impxxx__DATALIST.Rdata`
Saved object of a list containing multiply imputed datasets

`impxxx__IMP_LIST`
File with list of multiply imputed datasets

`impxxx__IMP_SUMMARY`
Summary file of the imputation

`impxxx__IMPDATA_nn`
Imputed datasets `nn`

`impxxx__IMPMETHOD`
File containing imputation methods

`impxxx__LEGENDE`
File with variable names of the dataset

`impxxx__LONG` Imputed datasets in long format

`impxxx__PREDICTORMATRIX`
File containing the predictor matrix

`impxxx__SPSS.sps`
SPSS syntax for reading the corresponding `txt` file into SPSS format.

Author(s)

Alexander Robitzsch

See Also

See also `mice::mids2mplus` and `mice::mids2spss`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Imputation of nhanes data and write imputed datasets on disk
#############################################################################

data(nhanes,package="mice")

#**********
# Model 1: Imputation using mice
imp1 <- mice( nhanes , m=3 , maxit=5 )
# write results
write.mice.imputation(mi.res=imp1 , name="mice_imp1" )

#**********
# Model 2: Imputation using mice.1chain
imp2 <- mice.1chain( nhanes , burnin=10 , iter=20 , Nimp=4 )
```

```
# write results
write.mice.imputation(mi.res=imp2 , name="mice_imp2" )

## End(Not run)
```

`write.pspp` *Writing a Data Frame into SPSS Format Using PSPP Software*

Description

Writes a data frame into SPSS format using the *PSPP* software. To use this function, download and install PSPP at first: http://www.gnu.org/software/pspp/pspp.html.

Usage

```
write.pspp(data, datafile, pspp.path, decmax = 6,
   as.factors=TRUE , use.bat=FALSE)
```

Arguments

`data`	Data frame
`datafile`	Name of the output file (without file ending)
`pspp.path`	Path where the PSPP executable is located, e.g. `"C:/Program Files (x86)/PSPP/bin/"`
`decmax`	Maximum number of digits after decimal
`as.factors`	A logical indicating whether all factors and string entries should be treated as factors in the output file.
`use.bat`	A logical indicating whether PSPP executed via a batch file in the DOS mode (`TRUE`) or directly envoked via the `system` command from within R (`FALSE`).

Value

A dataset in *sav* format (SPSS format).

Author(s)

Alexander Robitzsch

The code is adapted from https://stat.ethz.ch/pipermail/r-help/2006-January/085941.html

See Also

See also `foreign::write.foreign`.

For writing sav files see also `haven::write_sav` and `sjmisc::write_spss`.

For convenient viewing *sav* files we recommend the freeware program *ViewSav*, see http://www.asselberghs.dds.nl/stuff.htm.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Write a data frame into SPSS format
#############################################################################

#****
# (1) define data frame
data <- data.frame( "pid" = 1000+1:5 , "height" = round(rnorm( 5 ),4) ,
                "y" = 10*c(1,1,1,2,2) , "r2" = round( rnorm(5),2) ,
                "land" = as.factor( c( rep("A" ,1) , rep("B" , 4 ) ) ) )
#****
# (2) define variable labels
v1 <- rep( "" , ncol(data) )
names(v1) <-  colnames(data)
attr( data , "variable.labels" ) <- v1
attr(data,"variable.labels")["pid"] <- "Person ID"
attr(data,"variable.labels")["height"] <- "Height of a person"
attr(data,"variable.labels")["y"] <- "Gender"

#****
# (3) define some value labels
v1 <- c(10,20)
names(v1) <- c("male" , "female" )
attr( data$y , "value.labels" ) <- v1

#****
# (4a) run PSPP to produce a sav file
write.pspp( data , datafile = "example_data1" ,
        pspp.path = "C:/Program Files (x86)/PSPP/bin/" )

#****
# (4b) produce strings instead of factors
write.pspp( data , datafile = "example_data2" ,
        pspp.path = "C:/Program Files (x86)/PSPP/bin/"  , as.factors=FALSE )

#****
# write sav file using the haven package
library(haven)
haven::write_sav( data , "example_data1a.sav" )

#****
# write sav file using sjmisc package
data <- sjmisc::set_var_labels( data, attr(data, "variable.labels") )
sjmisc::write_spss( data , "example_data1b.sav" )

## End(Not run)
```

Index