Package 'mice'

July 2, 2014

Type Package

Version 2.22

Title Multivariate Imputation by Chained Equations

Date 2014-06-10

Maintainer Stef van Buuren <stef.vanbuuren@tno.nl>

Depends R (>= 2.10.0), methods, Rcpp (>= 0.10.6), lattice

Imports MASS, nnet, randomForest, rpart

Suggests AGD, gamlss, lme4, mitools, nlme, pan, survival, Zelig

LinkingTo Rcpp

Description Multiple imputation using Fully Conditional Specification (FCS) implemented by the MICE algorithm. Each variable has its own imputation model. Built-in imputation models are provided for continuous data (predictive mean matching, normal), binary data (logistic regression),unordered categorical data (polytomous logistic regression) and ordered categorical data (proportional odds). MICE can also impute continuous two-level data (normal model, pan, second-level variables). Passive imputation can be used to maintain consistency between variables. Various diagnostic plots are available to inspect the quality of the imputations.

License GPL-2 | GPL-3

LazyLoad yes

LazyData yes

URL http://www.stefvanbuuren.nl , http://www.multiple-imputation.com

Author Stef van Buuren [aut, cre],Karin Groothuis-Oudshoorn [aut],Alexander Robitzsch [ctb],Gerko Vink [ctb],Lisa Doove [ctb],Shahab Jolani [ctb]

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-06-11 23:31:32

R topics documented:

appendbreak *Appends specified break to the data*

Description

A custom function to insert rows in long data with new pseudo-observations that are being done on the specified break ages. There should be a column called `first` in `data` with logical data that codes whether the current row is the first for subject `id`. Furthermore, the function assumes that columns `age`, `occ`, `hgt.z`, `wgt.z` and `bmi.z` are available. This function is used on the `tbc` data in FIMD chapter 9. Check that out to see it in action.

Usage

```
appendbreak(data, brk, warp.model = warp.model, id = NULL, typ = "pred")
```

Arguments

`data`	A data frame in the long long format
`brk`	A vector of break ages
`warp.model`	A time warping model
`id`	The subject identifier
`typ`	Label to signal that this is a newly added observation

Value

A long data frame with additional rows for the break ages

as.mids *Converts an multiply imputed dataset (long format) into a* `mids` *object*

Description

This function converts imputed data stored in long format into an object of class `mids`. The original incomplete data set needs to be available so that we know where the missing data are. The function is useful to convert back operations applied to the imputed data back in a `mids` object. It may also be used to store multiply imputed data sets from other software into the format used by `mice`.

Usage

```
as.mids(data, .imp=1, .id=2)
```

Arguments

data	A multiply imputed data set in long format
.imp	Mandatory column indicator for the multiple imputation stream, where 0 indicates the incomplete data and 1 through m indicate the m multiple imputation streams. Default is 1.
.id	Optional column indicator for the row numbers. Default is 2.

Details

If .id is specified, row names from the original data (if supplied) will be copied to the mids object.

Value

An object of class mids

Author(s)

Gerko Vink, 2012

Examples

```
# nhanes example without .id
imp <- mice(nhanes, print = FALSE)
X <- complete(imp, action = "long", include = TRUE)[, -2]
test <- as.mids(X, .id = NULL)
is.mids(test)
test.dat <- complete(test, action = "long", include = TRUE)

# Test on boys data
imp <- mice(boys, print = FALSE, maxit = 1)
X <- complete(imp, action = "long", include = TRUE)
test <- as.mids(X)
is.mids(test)
test.dat <- complete(test, action = "long", include = TRUE)
# original rownumbers are automatically copied from .id
```

---

as.mira *Create a* mira *object from repeated analyses*

Description

The as.mira() function takes the results of repeated complete-data analysis stored as a list, and turns it into a mira object that can be pooled. Pooling requires that coef() and vcov() methods are available for fitted object.

Usage

```
as.mira(fitlist)
```

Arguments

fitlist	A list containing m fitted analysis objects

Value

An S3 object of class mira.

Author(s)

Stef van Buuren, 2011

See Also

mira

---

boys *Growth of Dutch boys*

Description

Height, weight, head circumference and puberty of 748 Dutch boys.

Format

A data frame with 748 rows on the following 9 variables:

age Decimal age (0-21 years)

hgt Height (cm)

wgt Weight (kg)

bmi Body mass index

hc Head circumference (cm)

gen Genital Tanner stage (G1-G5)

phb Pubic hair (Tanner P1-P6)

tv Testicular volume (ml)

reg Region (north, east, west, south, city)

Details

Random sample of 10% from the cross-sectional data used to construct the Dutch growth references 1997. Variables gen and phb are ordered factors. reg is a factor.

Source

Fredriks, A.M,, van Buuren, S., Burgmeijer, R.J., Meulmeester JF, Beuker, R.J., Brugman, E., Roede, M.J., Verloove-Vanhorick, S.P., Wit, J.M. (2000) Continuing positive secular growth change in The Netherlands 1955-1997. *Pediatric Research*, **47**, 316-323. http://www.stefvanbuuren.nl/publications/Continuingsecular-PedRes2000.pdf

Fredriks, A.M., van Buuren, S., Wit, J.M., Verloove-Vanhorick, S.P. (2000). Body index measurements in 1996-7 compared with 1980. *Archives of Disease in Childhood*, **82**, 107-112. http://www.stefvanbuuren.nl/publications/Bodyindex-ADC2000.pdf

Examples

```
# create two imputed data sets
imp <- mice(boys, m=1, maxit=2)
z <- complete(imp, 1)

# create imputations for age <8yrs
plot(z$age, z$gen, col=mdc(1:2)[1+is.na(boys$gen)],
 xlab = "Age (years)", ylab = "Tanner Stage Genital")

# figure to show that the default imputation method does not impute BMI
# consistently
plot(z$bmi,z$wgt/(z$hgt/100)^2, col=mdc(1:2)[1+is.na(boys$bmi)],
xlab = "Imputed BMI", ylab="Calculated BMI")

# also, BMI distributions are somewhat different
require(MASS)
oldpar <- par(mfrow=c(1,2))
truehist(z$bmi[!is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,
col=mdc(1),xlab="BMI observed")
truehist(z$bmi[is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,
col=mdc(2),xlab="BMI imputed")
par(oldpar)

# repair the inconsistency problem by passive imputation
meth <- imp$meth
meth["bmi"] <- "~I(wgt/(hgt/100)^2)"
pred <- imp$predictorMatrix
pred["hgt","bmi"] <- 0
pred["wgt","bmi"] <- 0
imp2 <- mice(boys, m=1, maxit=2, meth=meth, pred=pred)
z2 <- complete(imp2, 1)

# show that new imputations are consistent
plot(z2$bmi,z2$wgt/(z2$hgt/100)^2, col=mdc(1:2)[1+is.na(boys$bmi)],
ylab="Calculated BMI")

# and compare distributions
oldpar <- par(mfrow=c(1,2))
truehist(z2$bmi[!is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,col=mdc(1),
xlab="BMI observed")
truehist(z2$bmi[is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,col=mdc(2),
```

```
xlab="BMI imputed")
par(oldpar)
```

`bwplot.mids` *Box-and-whisker plot of observed and imputed data*

Description

Plotting methods for imputed data using **lattice**. `bwplot` produces box-and-whisker plots. The function automatically separates the observed and imputed data. The functions extend the usual features of **lattice**.

Usage

```
## S3 method for class 'mids'
bwplot(x, data, na.groups = NULL, groups = NULL,
  as.table = TRUE, theme = mice.theme(), mayreplicate = TRUE,
  allow.multiple = TRUE, outer = TRUE,
  drop.unused.levels = lattice.getOption("drop.unused.levels"), ...,
  subscripts = TRUE, subset = TRUE)
```

Arguments

`x`	A `mids` object, typically created by `mice()` or `mice.mids()`.
`data`	Formula that selects the data to be plotted. This argument follows the **lattice** rules for *formulas*, describing the primary variables (used for the per-panel display) and the optional conditioning variables (which define the subsets plotted in different panels) to be used in the plot. The formula is evaluated on the complete data set in the `long` form. Legal variable names for the formula include `names(x$data)` plus the two administrative factors `.imp` and `.id`. **Extended formula interface:** The primary variable terms (both the LHS `y` and RHS `x`) may consist of multiple terms separated by a '+' sign, e.g., `y1 + y2 ~ x | a * b`. This formula would be taken to mean that the user wants to plot both `y1 ~ x | a * b` and `y2 ~ x | a * b`, but with the `y1 ~ x` and `y2 ~ x` in *separate panels*. This behavior differs from standard **lattice**. *Only combine terms of the same type*, i.e. only factors or only numerical variables. Mixing numerical and categorical data occasionally produces odds labeling of vertical axis. For convience, in `stripplot()` and `bwplot` the formula `y~.imp` may be abbreviated as `y`. This applies only to a single `y`, and does not (yet) work for `y1+y2~.imp`.
`na.groups`	An expression evaluating to a logical vector indicating which two groups are distinguished (e.g. using different colors) in the display. The environment in which this expression is evaluated in the response indicator `is.na(x$data)`. The default `na.group = NULL` constrasts the observed and missing data in the LHS `y` variable of the display, i.e. groups created by `is.na(y)`. The expression `y` creates the groups according to `is.na(y)`. The expression `y1 & y2`

creates groups by `is.na(y1) & is.na(y2)`, and `y1 | y2` creates groups as `is.na(y1) | is.na(y2)`, and so on.

`groups` This is the usual `groups` arguments in **lattice**. It differs from `na.groups` because it evaluates in the completed data `data.frame(complete(x, "long", inc=TRUE))` (as usual), whereas `na.groups` evaluates in the response indicator. See `xyplot` for more details. When both `na.groups` and `groups` are specified, `na.groups` takes precedence, and `groups` is ignored.

`theme` A named list containing the graphical parameters. The default function `mice.theme` produces a short list of default colors, line width, and so on. The extensive list may be obtained from `trellis.par.get()`. Global graphical parameters like `col` or `cex` in high-level calls are still honored, so first experiment with the global parameters. Many setting consists of a pair. For example, `mice.theme` defines two symbol colors. The first is for the observed data, the second for the imputed data. The theme settings only exist during the call, and do not affect the trellis graphical parameters.

`mayreplicate` A logical indicating whether color, line widths, and so on, may be replicated. The graphical functions attempt to choose "intelligent" graphical parameters. For example, the same color can be replicated for different element, e.g. use all reds for the imputed data. Replication may be switched off by setting the flag to `FALSE`, in order to allow the user to gain full control.

`as.table` See `xyplot`.

`outer` See `xyplot`.

`allow.multiple` See `xyplot`.

`drop.unused.levels`
See `xyplot`.

`subscripts` See `xyplot`.

`subset` See `xyplot`.

`...` Further arguments, usually not directly processed by the high-level functions documented here, but instead passed on to other functions.

Details

The argument `na.groups` may be used to specify (combinations of) missingness in any of the variables. The argument `groups` can be used to specify groups based on the variable values themselves. Only one of both may be active at the same time. When both are specified, `na.groups` takes precedence over `groups`.

Use the `subset` and `na.groups` together to plots parts of the data. For example, select the first imputed data set by by `subset=.imp==1`.

Graphical paramaters like `col`, `pch` and `cex` can be specified in the arguments list to alter the plotting symbols. If `length(col)==2`, the color specification to define the observed and missing groups. `col[1]` is the color of the 'observed' data, `col[2]` is the color of the missing or imputed data. A convenient color choice is `col=mdc(1:2)`, a transparent blue color for the observed data, and a transparent red color for the imputed data. A good choice is `col=mdc(1:2), pch=20, cex=1.5`. These choices can be set for the duration of the session by running `mice.theme()`.

Value

The high-level functions documented here, as well as other high-level Lattice functions, return an object of class "trellis". The update method can be used to subsequently update components of the object, and the print method (usually called by default) will plot it on an appropriate plotting device.

Note

The first two arguments (x and data) are reversed compared to the standard Trellis syntax implemented in **lattice**. This reversal was necessary in order to benefit from automatic method dispatch.

In **mice** the argument x is always a mids object, whereas in **lattice** the argument x is always a formula.

In **mice** the argument data is always a formula object, whereas in **lattice** the argument data is usually a data frame.

All other arguments have identical interpretation.

Author(s)

Stef van Buuren

References

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*, Springer. http://lmdvr.r-forge.r-project.org/

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mice, xyplot, densityplot, stripplot, Lattice for an overview of the package, as well as bwplot, panel.bwplot, print.trellis, trellis.par.set

Examples

```
require(lattice)

imp <- mice(boys, maxit=1)

### box-and-whisker plot per imputation of all numerical variables
bwplot(imp)

### tv (testicular volume), conditional on region
bwplot(imp, tv~.imp|reg)

### same data, organized in a different way
bwplot(imp, tv~reg|.imp, theme=list())
```

cbind.mids *Columnwise combination of a* mids *object.*

Description

This function combines two mids objects columnwise into a single object of class mids, or combines a mids object with a vector, matrix, factor or data.frame columnwise into an object of class mids. The number of rows in the (incomplete) data x$data and y (or y$data if y is a mids object) should be equal. If y is a mids object then the number of imputations in x and y should be equal. Note: If y is a vector or factor its original name is lost and it will be denoted with y in the mids object.

Usage

```
cbind.mids(x, y, ...)
```

Arguments

x	A mids object.
y	A mids object or a data.frame, matrix, factor or vector.
...	Additional data.frame, matrix, vector or factor. These can be given as named arguments.

Value

An S3 object of class mids

Note

Component call is a vector, with first argument the mice() statement that created x and second argument the call to cbind.mids(). Component data is the codecbind of the (incomplete) data in x$data and y$data. Component m is the number of imputations. Component nmis is an array containing the number of missing observations per column. Component imp is a list of nvar components with the generated multiple imputations. Each part of the list is a nmis[j] by m matrix of imputed values for variable j. The original data of y will be copied into this list, including the missing values of y then y is not imputed. Component method is a vector of strings of length(nvar) specifying the elementary imputation method per column. If y is a mids object this vector is a combination of x$method and y$method, otherwise this vector is x$method and for the columns of y the method is set to ''. Component predictorMatrix is a square matrix of size ncol(data) containing integer data specifying the predictor set. If x and y are mids objects then the predictor matrices of x and y are combined with zero matrices on the off-diagonal blocks. Otherwise the variables in y are included in the predictor matrix of x such that y is not used as predictor(s) and not imputed as well. Component visitSequence is the sequence in which columns are visited. The same as x$visitSequence. Component seed is the seed value of the solution, x$seed. Component iteration is the last Gibbs sampling iteration number, x$iteration. Component lastSeedValue is the most recent seed value, x$lastSeedValue Component chainMean is the combination of x$chainMean and y$chainMean. If y$chainMean does not exist this element equals x$chainMean.

Component chainVar is the combination of x$chainVar and y$chainVar. If y$chainVar does not exist this element equals x$chainVar. Component pad is a list containing various settings of the padded imputation model, i.e. the imputation model after creating dummy variables. This list is defined by combining x$pad and y$pad if y is a mids object. Otherwise, it is defined by the settings of x and the combination of the data x$data and y. Component loggedEvents is set to x$loggedEvents. If a column of y is categorical this is ignored in the padded model since that column is not used as predictor for another column.

Author(s)

Karin Groothuis-Oudshoorn, Stef van Buuren, 2009

See Also

rbind.mids, ibind, mids

Examples

```
# append 'forgotten' variable bmi to imp
temp <- boys[,c(1:3,5:9)]
imp  <- mice(temp,maxit=1,m=2)
imp2 <- cbind.mids(imp, data.frame(bmi=boys$bmi))

# append maturation score to imp (numerical)
mat  <- (as.integer(temp$gen) + as.integer(temp$phb)
+ as.integer(cut(temp$tv,breaks=c(0,3,6,10,15,20,25))))
imp2 <- cbind.mids(imp, as.data.frame(mat))

# append maturation score to imp (factor)
# known issue: new column name is 'y', not 'mat'
mat  <- as.factor(mat)
imp2 <- cbind.mids(imp, mat)

# append data frame with two columns to imp
temp2 <- data.frame(bmi=boys$bmi,mat=as.factor(mat))
imp2  <- cbind.mids(imp, temp2)

# combine two mids objects
impa <- mice(temp, maxit=1, m=2)
impb <- mice(temp2, maxit=2, m=2)

# first a then b
impab <- cbind.mids(impa, impb)

# first b then a
impba <- cbind.mids(impb, impa)
```

cc *Complete cases*

Description

Extracting complete cases is also known as 'listwise deletion' or 'complete case analyses'. `cc(x)` is equivalent to `na.omit(x)`. Missing values in `x` are coded as `NA`. The companion function for selecting the incomplete cases is `ic()`.

Usage

```
cc(x, drop = TRUE)
```

Arguments

`x`	An `R` object. Currently supported are methods for the following classes: `mids`, `mira`, `mipo`, `data.frame` and `matrix`. In addition, `x` can be a vector of any kind.
`drop`	A logical flag for matrices and arrays. If `drop=TRUE` the result is coerced to the lowest possible dimension.

Value

A `vector`, `matrix` of `data.frame` containing the data of the complete cases.

Author(s)

Stef van Buuren, 2010.

See Also

`na.omit`, `ic`, `cci`, `ici`, `ccn`, `icn`

Examples

```
cc(nhanes)   # get the 13 complete cases
cc(nhanes[,2,drop=FALSE], drop=FALSE) # extract complete bmi as column
```

cci *Complete case indicator*

Description

This array is useful for extracting the subset of complete cases. The function `cci(x)` is equivelant to `complete.cases(x)`. Missing values in `x` are coded as `NA`. The companion function for selecting the incomplete cases is `ici()`.

Usage

```
cci(x)
```

Arguments

`x`	An `R` object. Currently supported are methods for the following classes: `mids`, `data.frame` and `matrix`. In addition, `x` can be a vector of any kind.

Value

Logical vector indicating the complete cases, . If `x` is a `data.frame` or `matrix` the length is `nrow(x)`. In other cases, the length is `length(x)`.

Author(s)

Stef van Buuren, 2010.

See Also

`complete.cases`, `ici` `cc`, `ic`, `ccn`, `icn`

Examples

```
cci(nhanes) # indicator for 13 complete cases
 f <- cci(nhanes[,c("bmi","hyp")]) # complete data for bmi and hyp
 nhanes[f,] # obtain all data from those with complete bmi and hyp
```

ccn *Complete cases n*

Description

Calculates the number of complete cases. The companion function for calculating the number of incomplete cases is `icn()`.

Usage

```
ccn(x)
```

Arguments

`x`	An `R` object. Currently supported are methods for the following classes: `mids`, `data.frame` and `matrix`. In addition, `x` can be a vector of any kind.

Value

An integer with the number of elements in `x` with complete data.

Author(s)

Stef van Buuren, 2010.

See Also

`icn`, `cc`, `ic`, `cci`, `ici`

Examples

```
ccn(nhanes) # 13 complete cases
```

complete *Creates imputed data sets from a* `mids` *object*

Description

Takes an object of class `mids`, fills in the missing data, and returns the completed data in a specified format.

Usage

```
complete(x, action = 1, include = FALSE)
```

Arguments

x	An object of class mids as created by the function mice().
action	If action is a scalar between 1 and x$m, the function returns the data with imputation number action filled in. Thus, action=1 returns the first completed data set, action=2 returns the second completed data set, and so on. The value of action can also be one of the following strings: 'long', 'broad', 'repeated'. See 'Details' for the interpretation.
include	Flag to indicate whether the orginal data with the missing values should be included. This requires that action is specified as 'long', 'broad' or 'repeated'.

Details

The argument action can also be a string, which is partially matched as follows:

list('\'long\'') produces a long data frame of vertically stacked imputed data sets with nrow(x$data) * x$m rows and ncol(x$data)+2 columns. The two additional columns are labeled .id containing the row names of x$data, and .imp containing the imputation number. If include=TRUE then nrow(x$data) additional rows with the original data are appended with .imp set equal to 0.

list('\'broad\'') produces a broad data frame with nrow(x$data) rows and ncol(x$data) * x$m columns. Columns are ordered such that the first ncol(x$data) columns corresponds to the first imputed data matrix. The imputation number is appended to each column name. If include=TRUE then ncol(x$data) additional columns with the original data are appended. The number .0 is appended to the column names.

list('\'repeated\'') produces a broad data frame with nrow(x$data) rows and ncol(x$data) * x$m columns. Columns are ordered such that the first x$m columns correspond to the x$m imputed versions of the first column in x$data. The imputation number is appended to each column name. If include=TRUE then ncol(x$data) additional columns with the original data are appended. The number .0 is appended to the column names.

Value

A data frame with the imputed values filled in. Optionally, the original data are appended.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2009

See Also

mice, mids

Examples

```
# do default multiple imputation on a numeric matrix
imp <- mice(nhanes)

# obtain first imputated matrix
mat <- complete(imp)
```

```
# fill in the third imputation
mat <- complete(imp, 3)

# long matrix with stacked complete data
mat <- complete(imp, 'long')

# long matrix with stacked complete data, including the original data
mat <- complete(imp, 'long', inc=TRUE)

# repeated matrix with complete data
mat <- complete(imp, 'r')

# for numeric data, produces a blocked correlation matrix, where
# each block contains of the same variable pair over different
# multiple imputations.
cor(mat)
```

`densityplot.mids` *Density plot of observed and imputed data*

Description

Plotting methods for imputed data using **lattice**. `densityplot` produces plots of the densities. The function automatically separates the observed and imputed data. The functions extend the usual features of **lattice**.

Usage

```
## S3 method for class 'mids'
densityplot(x, data, na.groups = NULL, groups = NULL,
  as.table = TRUE, plot.points = FALSE, theme = mice.theme(),
  mayreplicate = TRUE, thicker = 2.5, allow.multiple = TRUE,
  outer = TRUE,
  drop.unused.levels = lattice.getOption("drop.unused.levels"),
  panel = lattice.getOption("panel.densityplot"),
  default.prepanel = lattice.getOption("prepanel.default.densityplot"), ...,
  subscripts = TRUE, subset = TRUE)
```

Arguments

`x`	A `mids` object, typically created by `mice()` or `mice.mids()`.
`data`	Formula that selects the data to be plotted. This argument follows the **lattice** rules for *formulas*, describing the primary variables (used for the per-panel display) and the optional conditioning variables (which define the subsets plotted in different panels) to be used in the plot. The formula is evaluated on the complete data set in the `long` form. Legal variable names for the formula include `names(x$data)` plus the two administrative factors `.imp` and `.id`.

Extended formula interface: The primary variable terms (both the LHS `y` and RHS `x`) may consist of multiple terms separated by a '+' sign, e.g., `y1 + y2 ~ x | a * b`. This formula would be taken to mean that the user wants to plot both `y1 ~ x | a * b` and `y2 ~ x | a * b`, but with the `y1 ~ x` and `y2 ~ x` in *separate panels*. This behavior differs from standard **lattice**. *Only combine terms of the same type*, i.e. only factors or only numerical variables. Mixing numerical and categorical data occasionally produces odds labeling of vertical axis.

The function `densityplot` does not use the `y` terms in the formula. Density plots for `x1` and `x2` are requested as `~ x1 + x2`.

`na.groups` An expression evaluating to a logical vector indicating which two groups are distinguished (e.g. using different colors) in the display. The environment in which this expression is evaluated in the response indicator `is.na(x$data)`.

The default `na.group = NULL` constrasts the observed and missing data in the LHS `y` variable of the display, i.e. groups created by `is.na(y)`. The expression `y` creates the groups according to `is.na(y)`. The expression `y1 & y2` creates groups by `is.na(y1) & is.na(y2)`, and `y1 | y2` creates groups as `is.na(y1) | is.na(y2)`, and so on.

`groups` This is the usual `groups` arguments in **lattice**. It differs from `na.groups` because it evaluates in the completed data `data.frame(complete(x, "long", inc=TRUE))` (as usual), whereas `na.groups` evaluates in the response indicator. See `xyplot` for more details. When both `na.groups` and `groups` are specified, `na.groups` takes precedence, and `groups` is ignored.

`plot.points` A logical used in `densityplot` that signals whether the points should be plotted.

`theme` A named list containing the graphical parameters. The default function `mice.theme` produces a short list of default colors, line width, and so on. The extensive list may be obtained from `trellis.par.get()`. Global graphical parameters like `col` or `cex` in high-level calls are still honored, so first experiment with the global parameters. Many setting consists of a pair. For example, `mice.theme` defines two symbol colors. The first is for the observed data, the second for the imputed data. The theme settings only exist during the call, and do not affect the trellis graphical parameters.

`mayreplicate` A logical indicating whether color, line widths, and so on, may be replicated. The graphical functions attempt to choose "intelligent" graphical parameters. For example, the same color can be replicated for different element, e.g. use all reds for the imputed data. Replication may be switched off by setting the flag to `FALSE`, in order to allow the user to gain full control.

`thicker` Used in `densityplot`. Multiplication factor of the line width of the observed density. `thicker=1` uses the same thickness for the observed and imputed data.

`as.table` See `xyplot`.

`panel` See `xyplot`.

`default.prepanel` See `xyplot`.

`outer` See `xyplot`.

`allow.multiple` See `xyplot`.

drop.unused.levels
: See xyplot.

subscripts
: See xyplot.

subset
: See xyplot.

...
: Further arguments, usually not directly processed by the high-level functions documented here, but instead passed on to other functions.

Details

The argument na.groups may be used to specify (combinations of) missingness in any of the variables. The argument groups can be used to specify groups based on the variable values themselves. Only one of both may be active at the same time. When both are specified, na.groups takes precedence over groups.

Use the subset and na.groups together to plots parts of the data. For example, select the first imputed data set by by subset=.imp==1.

Graphical paramaters like col, pch and cex can be specified in the arguments list to alter the plotting symbols. If length(col)==2, the color specification to define the observed and missing groups. col[1] is the color of the 'observed' data, col[2] is the color of the missing or imputed data. A convenient color choice is col=mdc(1:2), a transparent blue color for the observed data, and a transparent red color for the imputed data. A good choice is col=mdc(1:2), pch=20, cex=1.5. These choices can be set for the duration of the session by running mice.theme().

Value

The high-level functions documented here, as well as other high-level Lattice functions, return an object of class "trellis". The update method can be used to subsequently update components of the object, and the print method (usually called by default) will plot it on an appropriate plotting device.

Note

The first two arguments (x and data) are reversed compared to the standard Trellis syntax implemented in **lattice**. This reversal was necessary in order to benefit from automatic method dispatch.

In **mice** the argument x is always a mids object, whereas in **lattice** the argument x is always a formula.

In **mice** the argument data is always a formula object, whereas in **lattice** the argument data is usually a data frame.

All other arguments have identical interpretation.

densityplot errs on empty groups, which occurs if all observations in the subgroup contain NA. The relevant error message is: Error in density.default: ... need at least 2 points to select a bandwidth automatically. There is yet no workaround for this problem. Use the more robust bwplot or stripplot as a replacement.

Author(s)

Stef van Buuren

References

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*, Springer. http://lmdvr.r-forge.r-project.org/

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mice, xyplot, stripplot, bwplot, Lattice for an overview of the package, as well as densityplot, panel.densityplot, print.trellis, trellis.par.set

Examples

```
require(lattice)

imp <- mice(boys, maxit=1)

### density plot of head circumference per imputation
### blue is observed, red is imputed
densityplot(imp, ~hc|.imp)

### All combined in one panel.
densityplot(imp, ~hc)
```

---

extractBS *Extract broken stick estimates from a* lmer *object*

Description

Extract broken stick estimates from a lmer object

Usage

```
extractBS(fit)
```

Arguments

fit	An object of class lmer

Value

A matrix containing broken stick estimates

Author(s)

Stef van Buuren, 2012

`fdd` *SE Fireworks disaster data*

Description

Multiple outcomes of a randomized study to reduce post-traumatic stress.

Format

`fdd` is a data frame with 52 rows and 65 columns:

id Client number

trt Treatment (E=EMDR, C=CBT)

pp Per protocol (Y/N)

trtp Number of parental treatments

sex Sex: M/F

etn Ethnicity: NL/OTHER

age Age (years)

trauma Trauma count (1-5)

prop1 PROPS total score T1

prop2 PROPS total score T2

prop3 PROPS total score T3

crop1 CROPS total score T1

crop2 CROPS total score T2

crop3 CROPS total score T3

masc1 MASC score T1

masc2 MASC score T2

masc3 MASC score T3

cbcl1 CBCL T1

cbcl3 CBCL T3

prs1 PRS total score T1

prs2 PRS total score T2

prs3 PRS total score T3

ypa1 PTSD-RI B intrusive recollection parent T1

ypb1 PTSD-RI C avoidant/numbing parent T1

ypc1 PTSD-RI D hyper-arousal parent T1

yp1 PTSD-RI B+C+D parent T1

ypa2 PTSD-RI B intrusive recollection parent T2

ypb2 PTSD-RI C avoidant/numbing parent T2

ypc2 PTSD-RI D hyper-arousal parent T2

yp2 PTSD-RI B+C+D parent T1

ypa3 PTSD-RI B intrusive recollection parent T3

ypb3 PTSD-RI C avoidant/numbing parent T3

ypc3 PTSD-RI D hyper-arousal parent T3

yp3 PTSD-RI B+C+D parent T3

yca1 PTSD-RI B intrusive recollection child T1

ycb1 PTSD-RI C avoidant/numbing child T1

ycc1 PTSD-RI D hyper-arousal child T1

yc1 PTSD-RI B+C+D child T1

yca2 PTSD-RI B intrusive recollection child T2

ycb2 PTSD-RI C avoidant/numbing child T2

ycc2 PTSD-RI D hyper-arousal child T2

yc2 PTSD-RI B+C+D child T2

yca3 PTSD-RI B intrusive recollection child T3

ycb3 PTSD-RI C avoidant/numbing child T3

ycc3 PTSD-RI D hyper-arousal child T3

yc3 PTSD-RI B+C+D child T3

ypf1 PTSD-RI parent full T1

ypf2 PTSD-RI parent full T2

ypf3 PTSD-RI parent full T3

ypp1 PTSD parent partial T1

ypp2 PTSD parent partial T2

ypp3 PTSD parent partial T3

ycf1 PTSD child full T1

ycf2 PTSD child full T2

ycf3 PTSD child full T3

ycp1 PTSD child partial T1

ycp2 PTSD child partial T2

ycp3 PTSD child partial T3

cbin1 CBCL Internalizing T1

cbin3 CBCL Internalizing T3

cbex1 CBCL Externalizing T1

cbex3 CBCL Externalizing T3

bir1 Birlison T1

bir2 Birlison T2

bir3 Birlison T3

`fdd.pred` is the 65 by 65 binary predictor matrix used to impute `fdd`.

Details

Data from a randomized experiment to reduce post-traumatic stress by two treatments: Eye Movement Desensitization and Reprocessing (EMDR) (experimental treatment), and cognitive behavioral therapy (CBT) (control treatment). 52 children were randomized to one of these two treatments. Outcomes were measured at three time points: at baseline (pre-treatment, T1), post-treatment (T2, 4-8 weeks), and at follow-up (T3, 3 months). For more details, see de Roos et al (2011). Some person covariates were reshuffled. The imputation methodology is explained in Chapter 9 of van Buuren (2012).

Source

de Roos, C., Greenwald, R., den Hollander-Gijsman, M., Noorthoorn, E., van Buuren, S., de Jong, A. (2011). A Randomised Comparison of Cognitive Behavioral Therapy (CBT) and Eye Movement Desensitisation and Reprocessing (EMDR) in disaster-exposed children. *European Journal of Psychotraumatology*, *2*, 5694.

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman \& Hall/CRC Press.

Examples

```
data <- fdd
md.pattern(fdd)
```

---

fdgs *Fifth Dutch growth study 2009*

Description

Age, height, weight and region of 10030 children measured within the Fifth Dutch Growth Study 2009

Format

`fdgs` is a data frame with 10030 rows and 8 columns:

id Person number

reg Region (factor, 5 levels)

age Age (years)

sex Sex (boy, girl)

hgt Height (cm)

wgt Weight (kg)

hgt.z Height Z-score

wgt.z Weight Z-score

Details

The data set contains data from children of Dutch descent (biological parents are born in the Netherlands). Children with growth-related diseases were excluded. The data were used to construct new growth charts of children of Dutch descent (Schonbeck 2013), and to calculate overweight and obesity prevalence (Schonbeck 2011).

Some groups were underrepresented. Multiple imputation was used to create synthetic cases that were used to correct for the nonresponse. See Van Buuren (2012), chapter 8 for details.

Source

Schonbeck, Y., Talma, H., van Dommelen, P., Bakker, B., Buitendijk, S. E., Hirasing, R. A., van Buuren, S. (2011). Increase in prevalence of overweight in Dutch children and adolescents: A comparison of nationwide growth studies in 1980, 1997 and 2009. *PLoS ONE*, *6*(11), e27608.

Schonbeck, Y., Talma, H., van Dommelen, P., Bakker, B., Buitendijk, S. E., Hirasing, R. A., \& van Buuren, S. (2013). The world's tallest nation has stopped growing taller: the height of Dutch children from 1955 to 2009. *Pediatric Research*, *73*(3), 371-377.

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Examples

```
data <- data(fdgs)
summary(data)
```

`fico` *Fraction of incomplete cases among cases with observed*

Description

FICO is an outbound statistic defined by the fraction of incomplete cases among cases with `Yj` observed (White and Carlin, 2010).

Usage

```
fico(data)
```

Arguments

`data`	A data frame or a matrix containing the incomplete data. Missing values are coded as NA's.

Value

A vector of length `ncol(data)` of FICO statistics.

Author(s)

Stef van Buuren, 2012

References

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

White, I.R., Carlin, J.B. (2010). Bias and efficiency of multiple imputation compared with complete-case analysis for missing covariate values. *Statistics in Medicine*, *29*, 2920-2931.

See Also

`fluxplot`, `flux`, `md.pattern`

`flux` *Influx and outflux of multivariate missing data patterns*

Description

Influx and outflux are statistics of the missing data pattern. These statistics are useful in selecting predictors that should go into the imputation model.

Usage

```
flux(data, local = names(data))
```

Arguments

`data`	A data frame or a matrix containing the incomplete data. Missing values are coded as NA's.
`local`	A vector of names of columns of `data`. The default is to include all columns in the calculations.

Details

Infux and outflux have been proposed by Van Buuren (2012), chapter 4.

Influx is equal to the number of variable pairs `(Yj , Yk)` with `Yj` missing and `Yk` observed, divided by the total number of observed data cells. Influx depends on the proportion of missing data of the variable. Influx of a completely observed variable is equal to 0, whereas for completely missing variables wehave influx = 1. For two variables with the same proportion of missing data, the variable with higher influx is better connected to the observed data, and might thus be easier to impute.

Outflux is equal to the number of variable pairs with `Yj` observed and `Yk` missing, divided by the total number of incomplete data cells. Outflux is an indicator of the potential usefulness of `Yj` for imputing other variables. Outflux depends on the proportion of missing data of the variable. Outflux of a completely observed variable is equal to 1, whereas outflux of a completely missing variable is equal to 0. For two variables having the same proportion of missing data, the variable with higher

outflux is better connected to the missing data, and thus potentially more useful for imputing other variables.

FICO is an outbound statistic defined by the fraction of incomplete cases among cases with `Yj` observed (White and Carlin, 2010).

Value

A data frame with `ncol(data)` rows and six columns: pobs = Proportion observed, influx = Influx outflux = Outflux ainb = Average inbound statistic aout = Averege outbound statistic fico = Fraction of incomplete cases among cases with `Yj` observed

Author(s)

Stef van Buuren, 2012

References

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

White, I.R., Carlin, J.B. (2010). Bias and efficiency of multiple imputation compared with complete-case analysis for missing covariate values. *Statistics in Medicine*, *29*, 2920-2931.

See Also

`fluxplot`, `md.pattern`, `fico`

`fluxplot` *Fluxplot of the missing data pattern*

Description

Influx and outflux are statistics of the missing data pattern. These statistics are useful in selecting predictors that should go into the imputation model.

Usage

```
fluxplot(data, local = names(data), plot = TRUE, labels = TRUE,
  xlim = c(0, 1), ylim = c(0, 1), las = 1, xlab = "Influx",
  ylab = "Outflux", main = paste("Influx-outflux pattern for",
  deparse(substitute(data))), eqscplot = TRUE, pty = "s", ...)
```

Arguments

data	A data frame or a matrix containing the incomplete data. Missing values are coded as NA's.
local	A vector of names of columns of data. The default is to include all columns in the calculations.
plot	Should a graph be produced?
labels	Should the points be labeled?
xlim	See par.
ylim	See par.
las	See par.
xlab	See par.
ylab	See par.
main	See par.
eqscplot	Should a square plot be produced?
pty	See par.
...	Further arguments passed to plot() or eqscplot().

Details

Infux and outflux have been proposed by Van Buuren (2012), chapter 4.

Influx is equal to the number of variable pairs (Yj , Yk) with Yj missing and Yk observed, divided by the total number of observed data cells. Influx depends on the proportion of missing data of the variable. Influx of a completely observed variable is equal to 0, whereas for completely missing variables wehave influx = 1. For two variables with the same proportion of missing data, the variable with higher influx is better connected to the observed data, and might thus be easier to impute.

Outflux is equal to the number of variable pairs with Yj observed and Yk missing, divided by the total number of incomplete data cells. Outflux is an indicator of the potential usefulness of Yj for imputing other variables. Outflux depends on the proportion of missing data of the variable. Outflux of a completely observed variable is equal to 1, whereas outflux of a completely missing variable is equal to 0. For two variables having the same proportion of missing data, the variable with higher outflux is better connected to the missing data, and thus potentially more useful for imputing other variables.

Value

An invisible data frame with ncol(data) rows and six columns: pobs = Proportion observed, influx = Influx outflux = Outflux ainb = Average inbound statistic aout = Averege outbound statistic fico = Fraction of incomplete cases among cases with Yj observed

Author(s)

Stef van Buuren, 2012

References

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

White, I.R., Carlin, J.B. (2010). Bias and efficiency of multiple imputation compared with complete-case analysis for missing covariate values. *Statistics in Medicine*, *29*, 2920-2931.

See Also

`flux`, `md.pattern`, `fico`

`getfit` *Extracts fit objects from* `mira` *object*

Description

`getfit` returns the list of objects containing the repeated analysis results, or optionally, one of these fit objects.

Usage

```
getfit(x, i = -1, simplify = FALSE)
```

Arguments

`x`	An object of class `mira`, typically produced by a call to `with()`.
`i`	An integer between 1 and `x$m` signalling the number of the repeated analysis. The default `i= -1` return a list with all analyses.
`simplify`	Should the return value be unlisted?

Details

This function is shorthand notation for `x$analyses` and `x$analyses[[i]]`.

Value

If `i = -1` an object containing all analyses, otherwise it returns the fittd object of the i'th repeated analysis.

Author(s)

Stef van Buuren, March 2012.

See Also

`mira`, `with.mids`

Examples

```
imp <- mice(nhanes)
fit <- with(imp, lm(bmi~chl+hyp))
getfit(fit)
getfit(fit, 2)
```

`glm.mids` *Generalized linear model for* `mids` *object*

Description

Applies `glm()` to a multiply imputed data set

Usage

```
glm.mids(formula, family = gaussian, data, ...)
```

Arguments

`formula`	a formula expression as for other regression models, of the form response ~ predictors. See the documentation of `lm` and `formula` for details.
`family`	The family of the glm model
`data`	An object of type `mids`, which stands for 'multiply imputed data set', typically created by function `mice()`.
`...`	Additional parameters passed to `glm`.

Details

This function is included for backward compatibility with V1.0. The function is superseeded by `with.mids`.

Value

An objects of class `mira`, which stands for 'multiply imputed repeated analysis'. This object contains `data$m` distinct `glm.objects`, plus some descriptive information.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, C.G.M. (2000) *Multivariate Imputation by Chained Equations: MICE V1.0 User's manual.* Leiden: TNO Quality of Life.

See Also

`with.mids`, `glm`, `mids`, `mira`

Examples

```
imp <- mice(nhanes)

# logistic regression on the imputed data
fit <- glm.mids((hyp==2)~bmi+chl, data=imp, family = binomial)
fit
```

`ibind` *Combine imputations fitted to the same data*

Description

This function combines two `mids` objects `x` and `y` into a single `mids` object. The two `mids` objects should have the same underlying multiple imputation model and should be fitted on exactly the same dataset. If the number of imputations in `x` is `m(x)` and in `y` is `m(y)` then the combination of both objects contains `m(x)+m(y)` imputations.

Usage

```
ibind(x, y)
```

Arguments

`x`	A `mids` object.
`y`	A `mids` object.

Value

An S3 object of class `mids`

Author(s)

Karin Groothuis-Oudshoorn, Stef van Buuren, 2009

See Also

`mids`, `rbind.mids`, `cbind.mids`

ic *Incomplete cases*

Description

Extracts incomplete cases from a data set. Missing values in x are coded as NA. The companion function for selecting the complete cases is cc().

Usage

```
ic(x, drop = TRUE)
```

Arguments

x	An R object. Currently supported are methods for the following classes: mids, mira, mipo, data.frame and matrix. In addition, x can be a vector of any kind.
drop	A logical flag for matrices and arrays. If drop=TRUE the result is coerced to the lowest possible dimension.

Value

A vector, matrix of data.frame containing the data of the incomplete cases.

Author(s)

Stef van Buuren, 2010.

See Also

na.omit, cc, cci, ici, ccn, icn

Examples

```
ic(nhanes)   # get the 12 rows with incomplete cases
ic(nhanes[1:10,])  # incomplete cases within the first ten rows
ic(nhanes[,2:3])  # restrict extraction to variables bmi and hyp
```

ici *Incomplete case indicator*

Description

This array is useful for extracting the subset of incomplete cases. Missing values in `x` are coded as `NA`. The companion function for selecting the complete cases is `cci()`.

Usage

```
ici(x)
```

Arguments

`x`	An `R` object. Currently supported are methods for the following classes: `mids`, `data.frame` and `matrix`. In addition, `x` can be a vector of any kind.

Value

Logical vector indicating the incomplete cases, . If `x` is a `data.frame` or `matrix` the length is `nrow(x)`. In other cases, the length is `length(x)`.

Author(s)

Stef van Buuren, 2010.

See Also

`cci` `cc`, `ic`, `ccn`, `icn`

Examples

```
ici(nhanes) # indicator for 12 rows with incomplete cases
```

icn *Incomplete cases n*

Description

Calculates the number of incomplete cases. The companion function for calculating the number of complete cases is `ccn()`.

Usage

```
icn(x)
```

Arguments

x	An R object. Currently supported are methods for the following classes: mids, data.frame and matrix. In addition, x can be a vector of any kind.

Value

An integer with the number of elements in x with incomplete data.

Author(s)

Stef van Buuren, 2010.

See Also

ccn, cc, ic, cci, ici

Examples

```
icn(nhanes) # the remaining 12 rows
 icn(nhanes[,c("bmi","hyp")]) # number of cases with incomplete bmi and hyp
```

---

is.mids *Check for* mids *object*

Description

Check for mids object

Usage

```
is.mids(x)
```

Arguments

x	An object

Value

A logical indicating whether x is an object of class mids

is.mipo *Check for* mipo *object*

Description

Check for mipo object

Usage

```
is.mipo(x)
```

Arguments

x	An object

Value

A logical indicating whether x is an object of class mipo

is.mira *Check for* mira *object*

Description

Check for mira object

Usage

```
is.mira(x)
```

Arguments

x	An object

Value

A logical indicating whether x is an object of class mira

leiden85 *Leiden 85+ study*

Description

Subset of data from the Leiden 85+ study

Format

`leiden85` is a data frame with 956 rows and 336 columns.

Details

The data set concerns of subset of 956 members of a very old (85+) cohort in Leiden.

Multiple imputation of this data set has been described in Boshuizen et al (1998), Van Buuren et al (1999) and Van Buuren (2012), chapter 7.

The data set is not available as part of `mice`.

Source

Lagaay, A. M., van der Meij, J. C., Hijmans, W. (1992). Validation of medical history taking as part of a population based survey in subjects aged 85 and over. *Brit. Med. J.*, *304*(6834), 1091-1092.

Izaks, G. J., van Houwelingen, H. C., Schreuder, G. M., Ligthart, G. J. (1997). The association between human leucocyte antigens (HLA) and mortality in community residents aged 85 and older. *Journal of the American Geriatrics Society*, *45*(1), 56-60.

Boshuizen, H. C., Izaks, G. J., van Buuren, S., Ligthart, G. J. (1998). Blood pressure and mortality in elderly people aged 85 and older: Community based study. *Brit. Med. J.*, *316*(7147), 1780-1784.

Van Buuren, S., Boshuizen, H.C., Knook, D.L. (1999) Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in Medicine*, **18**, 681–694.

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

lm.mids *Linear regression for* `mids` *object*

Description

Applies `lm()` to multiply imputed data set

Usage

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

Arguments

formula	a formula object, with the response on the left of a ~ operator, and the terms, separated by + operators, on the right. See the documentation of lm and formula for details.
data	An object of type 'mids', which stands for 'multiply imputed data set', typically created by a call to function mice().
...	Additional parameters passed to lm

Details

This function is included for backward compatibility with V1.0. The function is superseeded by with.mids.

Value

An objects of class mira, which stands for 'multiply imputed repeated analysis'. This object contains data$m distinct lm.objects, plus some descriptive information.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

lm, mids, mira

Examples

```
imp <- mice(nhanes)
fit <- lm.mids(bmi~hyp+chl, data = imp)
fit
```

---

mammalsleep *Mammal sleep data*

Description

Dataset from Allison and Cicchetti (1976) of 62 mammal species on the interrelationship between sleep, ecological, and constitutional variables. The dataset contains missing values on five variables.

Format

mammalsleep is a data frame with 62 rows and 11 columns:

species Species of animal

bw Body weight (kg)

brw Brain weight (g)

sws Slow wave ("nondreaming") sleep (hrs/day)

ps Paradoxical ("dreaming") sleep (hrs/day)

ts Total sleep (hrs/day) (sum of slow wave and paradoxical sleep)

mls Maximum life span (years)

gt Gestation time (days)

pi Predation index (1-5), 1 = least likely to be preyed upon

sei Sleep exposure index (1-5), 1 = least exposed (e.g. animal sleeps in a well-protected den), 5 = most exposed

odi Overall danger index (1-5) based on the above two indices and other information, 1 = least danger (from other animals), 5 = most danger (from other animals)

Details

Allison and Cicchetti (1976) investigated the interrelationship between sleep, ecological, and constitutional variables. They assessed these variables for 39 mammalian species. The authors concluded that slow-wave sleep is negatively associated with a factor related to body size. This suggests that large amounts of this sleep phase are disadvantageous in large species. Also, paradoxical sleep (REM sleep) was associated with a factor related to predatory danger, suggesting that large amounts of this sleep phase are disadvantageous in prey species.

Source

Allison, T., Cicchetti, D.V. (1976). Sleep in Mammals: Ecological and Constitutional Correlates. Science, 194(4266), 732-734.

Examples

```
sleep <- data(mammalsleep)
```

---

md.pairs *Missing data pattern by variable pairs*

Description

Number of observations per variable pair.

Usage

```
md.pairs(data)
```

Arguments

data	A data frame or a matrix containing the incomplete data. Missing values are coded as NA.

Details

The four components in the output value is have the following interpretation:

list('rr') response-response, both variables are observed

list('rm') response-missing, row observed, column missing

list('mr') missing -response, row missing, column observed

list('mm') missing -missing, both variables are missing

Value

A list of four components named rr, rm, mr and mm. Each component is square numerical matrix containing the number observations within four missing data pattern.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2009

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Examples

```
pat <- md.pairs(nhanes)
pat

# show that these four matrices decompose the total sample size
# for each pair
pat$rr + pat$rm + pat$mr + pat$mm

# percentage of usable cases to impute row variable from column variable
round(100*pat$mr/(pat$mr+pat$mm))
```

---

md.pattern *Missing data pattern*

Description

Display missing-data patterns.

Usage

```
md.pattern(x)
```

Arguments

x	A data frame or a matrix containing the incomplete data. Missing values are coded as NA's.

Details

This function is useful for investigating any structure of missing observation in the data. In specific case, the missing data pattern could be (nearly) monotone. Monotonicity can be used to simplify the imputation model. See Schafer (1997) for details. Also, the missing pattern could suggest which variables could potentially be useful for imputation of missing entries.

Value

A matrix with `ncol(x)+1` columns, in which each row corresponds to a missing data pattern (1=observed, 0=missing). Rows and columns are sorted in increasing amounts of missing information. The last column and row contain row and column counts, respectively.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Schafer, J.L. (1997), Analysis of multivariate incomplete data. London: Chapman&Hall.

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). `mice`: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Examples

```
md.pattern(nhanes)
#     age hyp bmi chl
#  13   1   1   1   1  0
#   1   1   1   0   1  1
#   3   1   1   1   0  1
#   1   1   0   0   1  2
#   7   1   0   0   0  3
#   0   8   9  10 27
```

mdc *Graphical parameter for missing data plots.*

Description

mdc returns colors used to distinguish observed, missing and combined data in plotting. mice.theme return a partial list of named objects that can be used as a theme in stripplot, bwplot, densityplot and xyplot.

Usage

```
mdc(r = "observed", s = "symbol", transparent = TRUE, cso = hcl(240,
  100, 40, 0.7), csi = hcl(0, 100, 40, 0.7), csc = "gray50",
  clo = hcl(240, 100, 40, 0.8), cli = hcl(0, 100, 40, 0.8),
  clc = "gray50")
```

Arguments

r	A numerical or character vector. The numbers 1-6 request colors as follows: 1=cso, 2=csi, 3=csc, 4=clo, 5=cli and 6=clc. Alternatively, r may contain the strings 'observed', 'missing', or 'both', or abbreviations thereof.
s	A character vector containing the strings 'symbol' or 'line', or abbreviations thereof.
transparent	A logical indicating whether alpha-transparancy is allowed. The default is TRUE.
cso	The symbol color for the observed data. The default is a transparent blue.
csi	The symbol color for the missing or imputed data. The default is a transparent red.
csc	The symbol color for the combined observed and imputed data. The default is a grey color.
clo	The line color for the observed data. The default is a slightly darker transparent blue.
cli	The line color for the missing or imputed data. The default is a slightly darker transparent red.
clc	The line color for the combined observed and imputed data. The default is a grey color.

Details

This function eases consistent use of colors in plots. The default follows the Abayomi convention, which uses blue for observed data, red for missing or imputed data, and black for combined data.

Value

mdc() returns a vector containing color definitions. The length of the output vector is calculate from the length of r and s. Elements of the input vectors are repeated if needed.

Author(s)

Stef van Buuren, sept 2012.

References

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*, Springer. http://lmdvr.r-forge.r-project.org/

See Also

hcl, rgb, xyplot.mids, xyplot, trellis.par.set

Examples

```
# all six colors
mdc(1:6)

# lines color for observed and missing data
mdc(c('obs','mis'), 'lin')
```

---

mice *Multivariate Imputation by Chained Equations (MICE)*

Description

Generates Multivariate Imputations by Chained Equations (MICE)

Usage

```
mice(data, m = 5, 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"), maxit = 5, diagnostics = TRUE,
  printFlag = TRUE, seed = NA, imputationMethod = NULL,
  defaultImputationMethod = NULL, data.init = NULL, ...)
```

Arguments

data	A data frame or a matrix containing the incomplete data. Missing values are coded as NA.
m	Number of multiple imputations. The default is m=5.

`method` Can be either a single string, or a vector of strings with length `ncol(data)`, specifying the elementary imputation method to be used for each column in data. If specified as a single string, the same method will be used for all columns. The default imputation method (when no argument is specified) depends on the measurement level of the target column and are specified by the `defaultMethod` argument. Columns that need not be imputed have the empty method `''`. See details for more information.

`predictorMatrix`
A square matrix of size `ncol(data)` containing 0/1 data specifying the set of predictors to be used for each target column. Rows correspond to target variables (i.e. variables to be imputed), in the sequence as they appear in data. A value of '1' means that the column variable is used as a predictor for the target variable (in the rows). The diagonal of `predictorMatrix` must be zero. The default for `predictorMatrix` is that all other columns are used as predictors (sometimes called massive imputation). Note: For two-level imputation codes '2' and '-2' are also allowed.

`visitSequence` A vector of integers of arbitrary length, specifying the column indices of the visiting sequence. The visiting sequence is the column order that is used to impute the data during one pass through the data. A column may be visited more than once. All incomplete columns that are used as predictors should be visited, or else the function will stop with an error. The default sequence `1:ncol(data)` implies that columns are imputed from left to right. It is possible to specify one of the keywords `'roman'` (left to right), `'arabic'` (right to left), `'monotone'` (sorted in increasing amount of missingness) and `'revmonotone'` (reverse of monotone). The keyword should be supplied as a string and may be abbreviated.

`post` A vector of strings with length `ncol(data)`, specifying expressions. Each string is parsed and executed within the `sampler()` function to postprocess imputed values. The default is to do nothing, indicated by a vector of empty strings `''`.

`form` A vector of strings with length `ncol(data)`, specifying formulae. Each string is parsed and executed within the `sampler()` function to create terms for the predictor. The default is to do nothing, indicated by a vector of empty strings `''`. The main value lies in the easy specification of interaction terms. The user must ensure that the set of variables in the formula match those in `predictors`.

`defaultMethod` A vector of three strings containing the default imputation methods for numerical columns, factor columns with 2 levels, and columns with (unordered or ordered) factors with more than two levels, respectively. If nothing is specified, the following defaults will be used: `pmm`, predictive mean matching (numeric data) `logreg`, logistic regression imputation (binary data, factor with 2 levels) `polyreg`, polytomous regression imputation for unordered categorical data (factor >= 2 levels) `polr`, proportional odds model for (ordered, >= 2 levels)

`maxit` A scalar giving the number of iterations. The default is 5.

`diagnostics` A Boolean flag. If `TRUE`, diagnostic information will be appended to the value of the function. If `FALSE`, only the imputed data are saved. The default is `TRUE`.

`printFlag` If `TRUE`, `mice` will print history on console. Use `print=FALSE` for silent computation.

`seed` An integer that is used as argument by the `set.seed()` for offsetting the random number generator. Default is to leave the random number generator alone.

`imputationMethod`
: Same as `method` argument. Included for backwards compatibility.

`defaultImputationMethod`
: Same as `defaultMethod` argument. Included for backwards compatibility.

`data.init`
: A data frame of the same size and type as `data`, without missing data, used to initialize imputations before the start of the iterative process. The default `NULL` implies that starting imputation are created by a simple random draw from the data. Note that specification of `data.init` will start the `m` Gibbs sampling streams from the same imputations.

`...`
: Named arguments that are passed down to the elementary imputation functions.

Details

Generates multiple imputations for incomplete multivariate data by Gibbs sampling. Missing data can occur anywhere in the data. The algorithm imputes an incomplete column (the target column) by generating 'plausible' synthetic values given other columns in the data. Each incomplete column must act as a target column, and has its own specific set of predictors. The default set of predictors for a given target consists of all other columns in the data. For predictors that are incomplete themselves, the most recently generated imputations are used to complete the predictors prior to imputation of the target column.

A separate univariate imputation model can be specified for each column. The default imputation method depends on the measurement level of the target column. In addition to these, several other methods are provided. You can also write their own imputation functions, and call these from within the algorithm.

The data may contain categorical variables that are used in a regressions on other variables. The algorithm creates dummy variables for the categories of these variables, and imputes these from the corresponding categorical variable. The extended model containing the dummy variables is called the padded model. Its structure is stored in the list component `pad`.

Built-in elementary imputation methods are:

pmm Predictive mean matching (any)

norm Bayesian linear regression (numeric)

norm.nob Linear regression ignoring model error (numeric)

norm.boot Linear regression using bootstrap (numeric)

norm.predict Linear regression, predicted values (numeric)

mean Unconditional mean imputation (numeric)

2l.norm Two-level normal imputation (numeric)

2l.pan Two-level normal imputation using pan (numeric)

2lonly.mean Imputation at level-2 of the class mean (numeric)

2lonly.norm Imputation at level-2 by Bayesian linear regression (numeric)

2lonly.pmm Imputation at level-2 by Predictive mean matching (any)

quadratic Imputation of quadratic terms (numeric)

logreg Logistic regression (factor, 2 levels)

logreg.boot Logistic regression with bootstrap

polyreg Polytomous logistic regression (factor, >= 2 levels)

polr Proportional odds model (ordered, >=2 levels)

lda Linear discriminant analysis (factor, >= 2 categories)

cart Classification and regression trees (any)

rf Random forest imputations (any)

ri Random indicator method for nonignorable data (numeric)

sample Random sample from the observed values (any)

fastpmm Experimental: Fast predictive mean matching using C++ (any)

These corresponding functions are coded in the `mice` library under names `mice.impute.method`, where `method` is a string with the name of the elementary imputation method name, for example `norm`. The `method` argument specifies the methods to be used. For the `j`'th column, `mice()` calls the first occurence of `paste('mice.impute.',method[j],sep='')` in the search path. The mechanism allows uses to write customized imputation function, `mice.impute.myfunc`. To call it for all columns specify `method='myfunc'`. To call it only for, say, column 2 specify `method=c('norm','myfunc','logreg',...{`

Passive imputation: `mice()` supports a special built-in method, called passive imputation. This method can be used to ensure that a data transform always depends on the most recently generated imputations. In some cases, an imputation model may need transformed data in addition to the original data (e.g. log, quadratic, recodes, interaction, sum scores, and so on).

Passive imputation maintains consistency among different transformations of the same data. Passive imputation is invoked if `~` is specified as the first character of the string that specifies the elementary method. `mice()` interprets the entire string, including the `~` character, as the formula argument in a call to `model.frame(formula,data[!r[,j],])`. This provides a simple mechanism for specifying determinstic dependencies among the columns. For example, suppose that the missing entries in variables `data$height` and `data$weight` are imputed. The body mass index (BMI) can be calculated within `mice` by specifying the string `'~I(weight/height^2)'` as the elementary imputation method for the target column `data$bmi`. Note that the `~` mechanism works only on those entries which have missing values in the target column. You should make sure that the combined observed and imputed parts of the target column make sense. An easy way to create consistency is by coding all entries in the target as `NA`, but for large data sets, this could be inefficient. Note that you may also need to adapt the default `predictorMatrix` to evade linear dependencies among the predictors that could cause errors like `Error in solve.default()` or `Error: system is exactly singular`. Though not strictly needed, it is often useful to specify `visitSequence` such that the column that is imputed by the `~` mechanism is visited each time after one of its predictors was visited. In that way, deterministic relation between columns will always be synchronized.

Value

Returns an S3 object of class `mids` (multiply imputed data set)

Author(s)

Stef van Buuren <stef.vanbuuren@tno.nl>, Karin Groothuis-Oudshoorn <c.g.m.oudshoorn@utwente.nl>, 2000-2010, with contributions of Alexander Robitzsch, Gerko Vink, Shahab Jolani, Roel de Jong, Jason Turner, Lisa Doove, John Fox, Frank E. Harrell, and Peter Malewski.

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Van Buuren, S., Brand, J.P.L., Groothuis-Oudshoorn C.G.M., Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, **76**, 12, 1049–1064.

Van Buuren, S. (2007) Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, **16**, 3, 219–242.

Van Buuren, S., Boshuizen, H.C., Knook, D.L. (1999) Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in Medicine*, **18**, 681–694.

Brand, J.P.L. (1999) *Development, implementation and evaluation of multiple imputation strategies for the statistical analysis of incomplete data sets.* Dissertation. Rotterdam: Erasmus University.

See Also

mids, with.mids, set.seed, complete

Examples

```
# do default multiple imputation on a numeric matrix
imp <- mice(nhanes)
imp

# list the actual imputations for BMI
imp$imputations$bmi

# first completed data matrix
complete(imp)


# imputation on mixed data with a different method per column

mice(nhanes2, meth=c('sample','pmm','logreg','norm'))
```

---

mice.impute.2l.norm *Imputation by a two-level normal model*

Description

Imputes univariate missing data using a two-level normal model

Usage

```
mice.impute.2l.norm(y, ry, x, type, intercept = 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	Vector of length ncol(x) identifying random and class variables. Random variables are identified by a '2'. The class variable (only one is allowed) is coded as '-2'. Random variables also include the fixed effect.
intercept	Logical determining whether the intercept is automatically added.
...	Other named arguments.

Details

Implements the Gibbs sampler for the linear multilevel model with heterogeneous with-class variance (Kasim and Raudenbush, 1998). Imputations are drawn as an extra step to the algorithm. For simulation work see Van Buuren (2011).

The random intercept is automatically added in `mice.impute.2L.norm()`. A model within a random intercept can be specified by `mice(...,intercept = FALSE)`.

Value

A vector of length nmis with imputations.

Note

Added June 25, 2012: The currently implemented algorithm does not handle predictors that are specified as fixed effects (type=1). When using `mice.impute.2l.norm()`, the current advice is to specify all predictors as random effects (type=2).

Warning: The assumption of heterogeneous variances requires that in every class at least one observation has a response in y.

Author(s)

Roel de Jong, 2008

References

Kasim RM, Raudenbush SW. (1998). Application of Gibbs sampling to nested variance components models with heterogeneous within-group variance. Journal of Educational and Behavioral Statistics, 23(2), 93–116.

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Van Buuren, S. (2011) Multiple imputation of multilevel data. In Hox, J.J. and and Roberts, J.K. (Eds.), *The Handbook of Advanced Multilevel Analysis*, Chapter 10, pp. 173–196. Milton Park, UK: Routledge.

mice.impute.2l.pan *Imputation by a two-level normal model using* pan

Description

Imputes univariate missing data using a two-level normal model with homogeneous within group variances. Aggregated group effects (i.e. group means) can be automatically created and included as predictors in the two-level regression (see argument type). This function needs the pan package.

Usage

```
mice.impute.2l.pan(y, ry, x, type, intercept = TRUE, paniter = 500,
  groupcenter.slope = 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.
type	Vector of length ncol(x) identifying random and class variables. Random effects are identified by a '2'. The group variable (only one is allowed) is coded as '-2'. Random effects also include the fixed effect. If for a covariates X1 group means shall be calculated and included as further fixed effects choose '3'. In addition to the effects in '3', specification '4' also includes random effects of X1.
intercept	Logical determining whether the intercept is automatically added.
paniter	Number of iterations in pan. Default is 500.
groupcenter.slope	If TRUE, in case of group means (type is '3' or'4') group mean centering for these predictors are conducted before doing imputations. Default is FALSE.
...	Other named arguments.

Details

Implements the Gibbs sampler for the linear two-level model with homogeneous within group variances which is a special case of a multivariate linear mixed effects model (Schafer & Yucel, 2002). For a two-level imputation with heterogeneous within-group variances see mice.impute.2l.norm. The random intercept is automatically added in mice.impute.2l.norm().

Value

A vector of length nmis with imputations.

Author(s)

Alexander Robitzsch (Federal Institute for Education Research, Innovation, and Development of the Austrian School System, Salzburg, Austria), <a.robitzsch@bifie.at>

References

Schafer J L, Yucel RM (2002). Computational strategies for multivariate linear mixed-effects models with missing values. *Journal of Computational and Graphical Statistics*. **11**, 437-457.

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). `mice`: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

`mice.impute.2l.norm`

Examples

```
####################################
# simulate some data
# two-level regression model with fixed slope

# number of groups
G <- 250
# number of persons
n <- 20
# regression parameter
beta <- .3
# intraclass correlation
rho <- .30
# correlation with missing response
rho.miss <- .10
# missing proportion
missrate <- .50
y1 <- rep( rnorm( G , sd = sqrt( rho ) ) , each=n ) + rnorm(G*n , sd = sqrt( 1 - rho ))
x <-  rnorm( G*n )
y <- y1 + beta  * x
dfr0 <- dfr <- data.frame( "group" = rep(1:G , each=n ) , "x" = x , "y" = y )
dfr[ rho.miss * x + rnorm( G*n , sd = sqrt( 1 - rho.miss ) ) < qnorm( missrate ) , "y" ] <- NA

#.....
# empty imputation in mice
imp0 <- mice( as.matrix(dfr)  , maxit=0 )
predM <- imp0$predictorMatrix
impM <- imp0$method

#...
# specify predictor matrix and imputationMethod
predM1 <- predM
predM1["y","group"] <- -2
predM1["y","x"] <- 1        # fixed x effects imputation
impM1 <- impM
```

```
impM1["y"] <- "2l.pan"

# multilevel imputation
imp1 <- mice( as.matrix( dfr ) , m = 1 , predictorMatrix = predM1 ,
           imputationMethod = impM1 , maxit=1 )
# multilevel analysis
library(lme4)
mod <- lmer( y ~ ( 1 + x | group) + x , data = complete(imp1) )
summary(mod)

##########################################
# Examples of predictorMatrix specification

# random x effects
# predM1["y","x"] <- 2

# fixed x effects and group mean of x
# predM1["y","x"] <- 3

# random x effects and group mean of x
# predM1["y","x"] <- 4
```

`mice.impute.2lonly.mean` *Imputation of the mean within the class*

Description

Imputes the mean of within the class

Usage

```
mice.impute.2lonly.mean(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`	Vector of length `ncol(x)` identifying random and class variables. The class variable (only one is allowed) is coded as '-2'.
`...`	Other named arguments.

Details

Observed values in `y` are averaged within the class, and replicated to the missing `y` within that class. If there are no observed data in the class, all entries of the class are set to `NaN`. This function is primarily useful for repairing incomplete data that are constant within the class, but that vary over the classes.

Value

A vector of length nmis with imputations.

Author(s)

Gerko Vink, Stef van Buuren, 2013

`mice.impute.2lonly.norm` *Imputation at level 2 by Bayesian linear regression*

Description

Imputes univariate missing data at level 2 using Bayesian linear regression analysis. Variables are level 1 are aggregated at level 2. The group identifier at level 2 must be indicated by `type=-2` in the `predictorMatrix`.

Usage

```
mice.impute.2lonly.norm(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.

Details

This function allows in combination with `mice.impute.2l.pan` switching regression imputation between level 1 and level 2 as described in Yucel (2008) or Gelman and Hill (2007, p. 541).

Value

A vector of length nmis with imputations.

Author(s)

Alexander Robitzsch (Federal Institute for Education Research, Innovation, and Development of the Austrian School System, Salzburg, Austria), <a.robitzsch@bifie.at>

References

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

Yucel, RM (2008). Multiple imputation inference for multivariate multilevel continuous data with ignorable non-response. *Philosophical Transactions of the Royal Society A*, **366**, 2389-2404.

See Also

mice.impute.norm, mice.impute.2lonly.pmm, mice.impute.2l.pan

Examples

```
#################################################
# simulate some data
# x,y ... level 1 variables
# v,w ... level 2 variables

G <- 250            # number of groups
n <- 20             # number of persons
beta <- .3          # regression coefficient
rho <- .30          # residual intraclass correlation
rho.miss <- .10     # correlation with missing response
missrate <- .50     # missing proportion
y1 <- rep( rnorm( G , sd = sqrt( rho ) ) , each=n ) + rnorm(G*n , sd = sqrt( 1 - rho ))
w <- rep( round( rnorm(G ) , 2 ) , each=n )
v <- rep( round( runif( G , 0 , 3 ) ) , each=n )
x <-  rnorm( G*n )
y <- y1 + beta  * x + .2 * w + .1 * v
dfr0 <- dfr <- data.frame( "group" = rep(1:G , each=n ) , "x" = x , "y" = y , "w" = w , "v" = v )
dfr[ rho.miss * x + rnorm( G*n , sd = sqrt( 1 - rho.miss ) ) < qnorm( missrate ) , "y" ] <- NA
dfr[ rep( rnorm(G) , each=n ) < qnorm( missrate ) , "w" ] <- NA
dfr[ rep( rnorm(G) , each=n ) < qnorm( missrate ) , "v" ] <- NA

#....
# empty mice imputation
imp0 <- mice( as.matrix(dfr)  , maxit=0 )
predM <- imp0$predictorMatrix
impM <- imp0$method

#...
# multilevel imputation
predM1 <- predM
predM1[c("w","y","v"),"group"] <- -2
predM1["y","x"] <- 1        # fixed x effects imputation
impM1 <- impM
impM1[c("y","w","v")] <- c("2l.pan" , "2lonly.norm" , "2lonly.pmm" )

# y ... imputation using pan
# w ... imputation at level 2 using norm
# v ... imputation at level 2 using pmm
```

```
imp1 <- mice( as.matrix( dfr ) , m = 1 , predictorMatrix = predM1 ,
          imputationMethod = impM1 , maxit=1 , paniter=500)
```

`mice.impute.2lonly.pmm` *Imputation at level 2 by predictive mean matching*

Description

Imputes univariate missing data at level 2 using predictive mean matching. Variables are level 1 are aggregated at level 2. The group identifier at level 2 must be indicated by `type=-2` in the `predictorMatrix`.

Usage

```
mice.impute.2lonly.pmm(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.

Details

This function allows in combination with `mice.impute.2l.pan` switching regression imputation between level 1 and level 2 as described in Yucel (2008) or Gelman and Hill (2007, p. 541).

Value

A vector of length `nmis` with imputations.

Author(s)

Alexander Robitzsch (Federal Institute for Education Research, Innovation, and Development of the Austrian School System, Salzburg, Austria), <a.robitzsch@bifie.at>

References

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

Yucel, RM (2008). Multiple imputation inference for multivariate multilevel continuous data with ignorable non-response. *Philosophical Transactions of the Royal Society A*, **366**, 2389-2404.

See Also

mice.impute.pmm, mice.impute.2lonly.norm, mice.impute.2l.pan

Examples

```
################################################
# simulate some data
# x,y ... level 1 variables
# v,w ... level 2 variables

G <- 250            # number of groups
n <- 20             # number of persons
beta <- .3          # regression coefficient
rho <- .30          # residual intraclass correlation
rho.miss <- .10     # correlation with missing response
missrate <- .50     # missing proportion
y1 <- rep( rnorm( G , sd = sqrt( rho ) ) , each=n ) + rnorm(G*n , sd = sqrt( 1 - rho ))
w <- rep( round( rnorm(G ) , 2 ) , each=n )
v <- rep( round( runif( G , 0 , 3 ) ) , each=n )
x <-  rnorm( G*n )
y <- y1 + beta  * x + .2 * w + .1 * v
dfr0 <- dfr <- data.frame( "group" = rep(1:G , each=n ) , "x" = x , "y" = y , "w" = w , "v" = v )
dfr[ rho.miss * x + rnorm( G*n , sd = sqrt( 1 - rho.miss ) ) < qnorm( missrate ) , "y" ] <- NA
dfr[ rep( rnorm(G) , each=n ) < qnorm( missrate ) , "w" ] <- NA
dfr[ rep( rnorm(G) , each=n ) < qnorm( missrate ) , "v" ] <- NA

#....
# empty mice imputation
imp0 <- mice( as.matrix(dfr)  , maxit=0 )
predM <- imp0$predictorMatrix
impM <- imp0$method

#...
# multilevel imputation
predM1 <- predM
predM1[c("w","y","v"),"group"] <- -2
predM1["y","x"] <- 1        # fixed x effects imputation
impM1 <- impM
impM1[c("y","w","v")] <- c("2l.pan" , "2lonly.norm" , "2lonly.pmm" )

# y ... imputation using pan
# w ... imputation at level 2 using norm
# v ... imputation at level 2 using pmm

imp1 <- mice( as.matrix( dfr ) , m = 1 , predictorMatrix = predM1 ,
           imputationMethod = impM1 , maxit=1 , paniter=500)
```

---

mice.impute.cart *Imputation by classification and regression trees*

Description

Imputes univariate missing data using classification and regression trees.

Usage

```
mice.impute.cart(y, ry, x, minbucket = 5, cp = 1e-04, ...)
```

Arguments

y	Numeric vector with incomplete data
ry	Response pattern of y (TRUE = observed, FALSE = missing)
x	Design matrix with length(y) rows and p columns containing complete covariates.
minbucket	The minimum number of observations in any terminal node used. See rpart.control for details.
cp	Complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. See rpart.control for details.
...	Other named arguments passed down to rpart().

Details

Imputation of y by classification and regression trees. The procedure is as follows:

1. Fit a classification or regression tree by recursive partitioning;
2. For each ymis, find the terminal node they end up according to the fitted tree;
3. Make a random draw among the member in the node, and take the observed value from that draw as the imputation.

Value

Numeric vector of length sum(!ry) with imputations

Author(s)

Lisa Doove, Stef van Buuren, Elise Dusseldorp, 2012

References

Doove, L.L., van Buuren, S., Dusseldorp, E. (2014), Recursive partitioning for missing data imputation in the presence of interaction Effects. Computational Statistics \& Data Analysis, 72, 92-104.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), Classification and regression trees, Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software.

Van Buuren, S.(2012), Flexible imputation of missing data, Boca Raton, FL: Chapman & Hall/CRC.

See Also

mice, mice.impute.rf, rpart, rpart.control

Examples

```
require(rpart)
require(lattice)

imp <- mice(nhanes2, meth = "cart", minbucket = 4)
plot(imp)
```

`mice.impute.fastpmm` *Imputation by fast predictive mean matching*

Description

Imputes univariate missing data using fast predictive mean matching

Usage

```
mice.impute.fastpmm(y, ry, x, donors = 5, type = 1, ridge = 1e-05,
  version = "", ...)
```

Arguments

`y`	Numeric vector with incomplete data
`ry`	Response pattern of `y` (`TRUE`=observed, `FALSE`=missing)
`x`	Design matrix with `length(y)` rows and `p` columns containing complete covariates.
`donors`	The size of the donor pool among which a draw is made. The default is `donors = 5`. Setting `donors = 1` always selects the closest match. Values between 3 and 10 provide the best results. Note: The default was changed from 3 to 5 in version 2.19, based on simulation work by Tim Morris.
`type`	Type of matching distance. The default choice `type = 1` calculates the distance between the predicted value of `yobs` and the drawn values of `ymis`. Other choices are `type = 0` (distance between predicted values) and `type = 2` (distance between drawn values). The current version supports only `type = 1`.
`ridge`	The ridge penalty applied in `.norm.draw()` to prevent problems with multicollinearity. The default is `ridge = 1e-05`, which means that 0.01 percent of the diagonal is added to the cross-product. Larger ridges may result in more biased estimates. For highly noisy data (e.g. many junk variables), set `ridge = 1e-06` or even lower to reduce bias. For highly collinear data, set `ridge = 1e-04` or higher.
`version`	A character variable indicating the version. Currently unused.
`...`	Other named arguments.

Details

Imputation of y by predictive mean matching, based on Rubin (1987, p. 168, formulas a and b). The procedure is as follows:

1. Estimate beta and sigma by linear regression
2. Draw beta* and sigma* from the proper posterior
3. Compute predicted values for yobs beta and ymis beta*
4. For each ymis, find donors observations with closest predicted values, randomly sample one of these, and take its observed value in y as the imputation.
5. Ties are broken by making a random draw among ties. Note: The matching is done on predicted y, NOT on observed y.

Value

Numeric vector of length sum(!ry) with imputations

Note

The mice.impute.fastpmm() function is an experimental version of the standard mice.impute.pmm() function. In mice 2.22 both are equivalent. In future versions of mice the mice.impute.fastpmm() function may be subject to additional optimizations. This is an experimental feature.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000, 2012

References

Little, R.J.A. (1988), Missing data adjustments in large surveys (with discussion), Journal of Business Economics and Statistics, 6, 287–301.

Rubin, D.B. (1987). Multiple imputation for nonresponse in surveys. New York: Wiley.

Van Buuren, S., Brand, J.P.L., Groothuis-Oudshoorn C.G.M., Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, **76**, 12, 1049–1064.

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mice.impute.pmm

`mice.impute.lda` *Imputation by linear discriminant analysis*

Description

Imputes univariate missing data using linear discriminant analysis

Usage

```
mice.impute.lda(y, ry, x, ...)
```

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.
`...`	Other named arguments.

Details

Imputation of categorical response variables by linear discriminant analysis. This function uses the Venables/Ripley functions `lda()` and `predict.lda()` to compute posterior probabilities for each incomplete case, and draws the imputations from this posterior.

Value

A vector of length `nmis` with imputations.

Warning

The function does not incorporate the variability of the discriminant weight, so it is not 'proper' in the sense of Rubin. For small samples and rare categories in the `y`, variability of the imputed data could therefore be somewhat underestimated.

Note

This function can be called from within the Gibbs sampler by specifying 'lda' in the `method` argument of `mice()`. This method is usually faster and uses fewer resources than calling the function `mice.impute.polyreg`.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999). Development, Implementation and Evaluation of Multiple Imputation Strategies for the Statistical Analysis of Incomplete Data Sets. Ph.D. Thesis, TNO Prevention and Health/Erasmus University Rotterdam. ISBN 90-74479-08-1.

Venables, W.N. & Ripley, B.D. (1997). Modern applied statistics with S-PLUS (2nd ed). Springer, Berlin.

See Also

mice, link{mice.impute.polyreg}, lda

---

mice.impute.logreg *Imputation by logistic regression*

Description

Imputes univariate missing data using logistic regression.

Usage

```
mice.impute.logreg(y, ry, x, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern of length n (FALSE=missing, TRUE=observed)
x	Matrix (n x p) of complete covariates.
...	Other named arguments.

Details

Imputation for binary response variables by the Bayesian logistic regression model (Rubin 1987, p. 169-170). The Bayesian method consists of the following steps:

1. Fit a logit, and find (bhat, V(bhat))
2. Draw BETA from N(bhat, V(bhat))
3. Compute predicted scores for m.d., i.e. logit-1(X BETA)
4. Compare the score to a random (0,1) deviate, and impute.

The method relies on the standard glm.fit function. Warnings from glm.fit are suppressed. Perfect prediction is handled by the data augmentation method.

Value

A vector of length nmis with imputations (0 or 1).

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000, 2011

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999). Development, Implementation and Evaluation of Multiple Imputation Strategies for the Statistical Analysis of Incomplete Data Sets. Ph.D. Thesis, TNO Prevention and Health/Erasmus University Rotterdam. ISBN 90-74479-08-1.

Venables, W.N. & Ripley, B.D. (1997). Modern applied statistics with S-Plus (2nd ed). Springer, Berlin.

White, I., Daniel, R. and Royston, P (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. Computational Statistics and Data Analysis, 54:22672275.

See Also

mice, glm, glm.fit

---

mice.impute.logreg.boot

Imputation by logistic regression using the bootstrap

Description

Imputes univariate missing data using logistic regression by a bootstrapped logistic regression model. The bootstrap method draws a simple bootstrap sample with replacement from the observed data y[ry] and x[ry,]. Perfect prediction is handled by the data augmentation method.

Usage

```
mice.impute.logreg.boot(y, ry, x, ...)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern of length n (FALSE=missing, TRUE=observed)
x	Matrix (n x p) of complete covariates.
...	Other named arguments.

Value

A vector of length nmis with imputations (0 or 1).

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000, 2011

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999). Development, Implementation and Evaluation of Multiple Imputation Strategies for the Statistical Analysis of Incomplete Data Sets. Ph.D. Thesis, TNO Prevention and Health/Erasmus University Rotterdam. ISBN 90-74479-08-1.

Venables, W.N. & Ripley, B.D. (1997). Modern applied statistics with S-Plus (2nd ed). Springer, Berlin.

White, I., Daniel, R. and Royston, P (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. Computational Statistics and Data Analysis, 54:22672275.

See Also

mice, glm, glm.fit

---

mice.impute.mean *Imputation by the mean*

Description

Imputes the arithmetic mean of the observed data

Usage

```
mice.impute.mean(y, ry, x = 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.
...	Other named arguments.

Value

A vector of length nmis with imputations.

Warning

Imputing the mean of a variable is almost never appropriate. See Little and Rubin (1987).

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Little, R.J.A. and Rubin, D.B. (1987). Statistical Analysis with Missing Data. New York: John Wiley and Sons.

See Also

mice, mean

---

mice.impute.norm *Imputation by Bayesian linear regression*

Description

Imputes univariate missing data using Bayesian linear regression analysis

Usage

```
mice.impute.norm(y, ry, x, ...)
```

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.
...	Other named arguments.

Details

Draws values of beta and sigma for Bayesian linear regression imputation of y given x according to Rubin p. 167.

Value

A vector of length nmis with imputations.

Note

Using `mice.impute.norm` for all columns is similar to Schafer's NORM method (Schafer, 1997).

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999) *Development, implementation and evaluation of multiple imputation strategies for the statistical analysis of incomplete data sets.* Dissertation. Rotterdam: Erasmus University.

Schafer, J.L. (1997). Analysis of incomplete multivariate data. London: Chapman & Hall.

`mice.impute.norm.boot` *Imputation by linear regression, bootstrap method*

Description

Imputes univariate missing data using linear regression with boostrap

Usage

```
mice.impute.norm.boot(y, ry, x, ridge = 1e-05, ...)
```

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
`...`	Other named arguments.

Details

Draws a bootstrap sample from `x[ry,]` and `y[ry]`, calculates regression weights and imputes with normal residuals. The `ridge` parameter adds a penalty term `ridge*diag(xtx)` to the variance-covariance matrix `xtx`.

Value

A vector of length `nmis` with imputations.

Author(s)

Stef van Buuren, 2011

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

---

mice.impute.norm.nob *Imputation by linear regression (non Bayesian)*

Description

Imputes univariate missing data using linear regression analysis (non Bayesian version)

Usage

```
mice.impute.norm.nob(y, ry, x, ...)
```

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.
...	Other named arguments.

Details

This creates imputation using the spread around the fitted linear regression line of y given x, as fitted on the observed data.

Value

A vector of length nmis with imputations.

Warning

The function does not incorporate the variability of the regression weights, so it is not 'proper' in the sense of Rubin. For small samples, variability of the imputed data is therefore underestimated.

Note

This function is provided mainly to allow comparison between proper and improper norm methods. Also, it may be useful to impute large data containing many rows.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999). Development, Implementation and Evaluation of Multiple Imputation Strategies for the Statistical Analysis of Incomplete Data Sets. Ph.D. Thesis, TNO Prevention and Health/Erasmus University Rotterdam.

See Also

mice, mice.impute.norm

---

mice.impute.norm.predict

Imputation by linear regression, prediction method

Description

Imputes univariate missing data using the predicted value from a linear regression

Usage

```
mice.impute.norm.predict(y, ry, x, ridge = 1e-05, ...)
```

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
...	Other named arguments.

Details

Calculates regression weights from the observed data and and return predicted values to as imputations. The ridge parameter adds a penalty term ridge*diag(xtx) to the variance-covariance matrix xtx.

Value

A vector of length nmis with imputations.

Author(s)

Stef van Buuren, 2011

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

---

mice.impute.passive *Passive imputation*

Description

Derive a new variable based on the imputed data

Usage

```
mice.impute.passive(data, func)
```

Arguments

data	A data frame
func	A formula specifying the transformations on data

Details

Passive imputation is a special internal imputation function. Using this facility, the user can specify, at any point in the mice Gibbs sampling algorithm, a function on the imputed data. This is useful, for example, to compute a cubic version of a variable, a transformation like `Q = W/H^2` based on two variables, or a mean variable like `(x_1+x_2+x_3)/3`. The so derived variables might be used in other places in the imputation model. The function allows to dynamically derive virtually any function of the imputed data at virtually any time.

Value

The result of applying formula

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mice

`mice.impute.pmm` *Imputation by predictive mean matching*

Description

Imputes univariate missing data using predictive mean matching

Usage

```
mice.impute.pmm(y, ry, x, donors = 5, type = 1, ridge = 1e-05,
  version = "", ...)
```

Arguments

`y`	Numeric vector with incomplete data
`ry`	Response pattern of `y` (`TRUE`=observed, `FALSE`=missing)
`x`	Design matrix with `length(y)` rows and `p` columns containing complete covariates.
`donors`	The size of the donor pool among which a draw is made. The default is `donors = 5`. Setting `donors = 1` always selects the closest match. Values between 3 and 10 provide the best results. Note: The default was changed from 3 to 5 in version 2.19, based on simulation work by Tim Morris.
`type`	Type of matching distance. The default choice `type = 1` calculates the distance between the predicted value of `yobs` and the drawn values of `ymis`. Other choices are `type = 0` (distance between predicted values) and `type = 2` (distance between drawn values). The current version supports only `type = 1`.
`ridge`	The ridge penalty applied in `.norm.draw()` to prevent problems with multicollinearity. The default is `ridge = 1e-05`, which means that 0.01 percent of the diagonal is added to the cross-product. Larger ridges may result in more biased estimates. For highly noisy data (e.g. many junk variables), set `ridge = 1e-06` or even lower to reduce bias. For highly collinear data, set `ridge = 1e-04` or higher.
`version`	A character variable indicating the version to be used. Specifying `version = "2.21"` calls `.pmm.match()` instead of the default `matcher()` function.
`...`	Other named arguments.

Details

Imputation of `y` by predictive mean matching, based on Rubin (1987, p. 168, formulas a and b). The procedure is as follows:

1. Estimate beta and sigma by linear regression
2. Draw beta* and sigma* from the proper posterior
3. Compute predicted values for `yobs` `beta` and `ymis` `beta*`

4. For each ymis, find donors observations with closest predicted values, randomly sample one of these, and take its observed value in y as the imputation.
5. Ties are broken by making a random draw among ties. Note: The matching is done on predicted y, NOT on observed y.

Value

Numeric vector of length sum(!ry) with imputations

Note

Since mice 2.22 the standard mice.impute.pmm() calls the much faster matcher() function instead of .pmm.match(). Since matcher() uses its own random generator, results cannot be exactly reproduced. In case where you want the old .pmm.match(), specify mice(..., version = "2.21").

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000, 2012

References

Little, R.J.A. (1988), Missing data adjustments in large surveys (with discussion), Journal of Business Economics and Statistics, 6, 287–301.

Rubin, D.B. (1987). Multiple imputation for nonresponse in surveys. New York: Wiley.

Van Buuren, S., Brand, J.P.L., Groothuis-Oudshoorn C.G.M., Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, **76**, 12, 1049–1064.

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

---

mice.impute.polr *Imputation by polytomous regression - ordered*

Description

Imputes missing data in a categorical variable using polytomous regression

Usage

```
mice.impute.polr(y, ry, x, nnet.maxit = 100, nnet.trace = FALSE,
  nnet.maxNWts = 1500, ...)
```

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.
nnet.maxit	Tuning parameter for nnet().
nnet.trace	Tuning parameter for nnet().
nnet.maxNWts	Tuning parameter for nnet().
...	Other named arguments.

Details

By default, ordered factors with more than two levels are imputed by mice.impute.polr.

The function mice.impute.polr() imputes for ordered categorical response variables by the proportional odds logistic regression (polr) model. The function repeatedly applies logistic regression on the successive splits. The model is also known as the cumulative link model.

The algorithm of mice.impute.polr uses the function polr() from the MASS package.

In order to avoid bias due to perfect prediction, the algorithm augment the data according to the method of White, Daniel and Royston (2010).

The call to polr might fail, usually because the data are very sparse. In that case, multinom is tried as a fallback, and a record is written to the loggedEvents component of the mids object.

Value

A vector of length nmis with imputations.

Author(s)

Stef van Buuren, Karin Groohuis-Oudshoorn, 2000-2010

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999) *Development, implementation and evaluation of multiple imputation strategies for the statistical analysis of incomplete data sets.* Dissertation. Rotterdam: Erasmus University.

White, I.R., Daniel, R. Royston, P. (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. *Computational Statistics and Data Analysis*, 54, 2267-2275.

Venables, W.N. & Ripley, B.D. (2002). *Modern applied statistics with S-Plus (4th ed)*. Springer, Berlin.

See Also

mice, multinom, polr

`mice.impute.polyreg` *Imputation by polytomous regression - unordered*

Description

Imputes missing data in a categorical variable using polytomous regression

Usage

```
mice.impute.polyreg(y, ry, x, nnet.maxit = 100, nnet.trace = FALSE,
  nnet.maxNWts = 1500, ...)
```

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.
`nnet.maxit`	Tuning parameter for `nnet()`.
`nnet.trace`	Tuning parameter for `nnet()`.
`nnet.maxNWts`	Tuning parameter for `nnet()`.
`...`	Other named arguments.

Details

By default, unordered factors with more than two levels are imputed by `mice.impute.polyreg()`. 'The function `mice.impute.polyreg()` imputes categorical response variables by the Bayesian polytomous regression model. See J.P.L. Brand (1999), Chapter 4, Appendix B.

The method consists of the following steps:

1. Fit categorical response as a multinomial model
2. Compute predicted categories
3. Add appropriate noise to predictions.

The algorithm of `mice.impute.polyreg` uses the function `multinom()` from the `nnet` package.

In order to avoid bias due to perfect prediction, the algorithm augment the data according to the method of White, Daniel and Royston (2010).

Value

A vector of length `nmis` with imputations.

Author(s)

Stef van Buuren, Karin Groohuis-Oudshoorn, 2000-2010

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Brand, J.P.L. (1999) *Development, implementation and evaluation of multiple imputation strategies for the statistical analysis of incomplete data sets.* Dissertation. Rotterdam: Erasmus University.

White, I.R., Daniel, R. Royston, P. (2010). Avoiding bias due to perfect prediction in multiple imputation of incomplete categorical variables. *Computational Statistics and Data Analysis*, 54, 2267-2275.

Venables, W.N. & Ripley, B.D. (2002). *Modern applied statistics with S-Plus (4th ed)*. Springer, Berlin.

See Also

mice, multinom, polr

---

mice.impute.quadratic *Imputation of quadratric terms*

Description

Imputes univariate missing data of incomplete variable that appears as both main effect and quadratic effect in the complete-data model.

Usage

```
mice.impute.quadratic(y, ry, x, ...)
```

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.
...	Other named arguments.

Details

This implements polynomial combination method. First, the polynomial combination $Z = Y beta_1 + Y^2 beta_2$ is formed. Z is imputed by predictive mean matching, followed by a decomposition of the imputed data Z into components Y and Y^2. See Van Buuren (2012, pp. 139-141) and Vink et al (2012) for more details. The method ensures that 1) the imputed data for Y and Y^2 are mutually consistent, and 2) that provides unbiased estimates of the regression weights in a complete-data linear regression that use both Y and Y^2.

Value

A vector of length nmis with imputations.

Note

There are two situations to consider. If only the linear term `Y` is present in the data, calculate the quadratic term `YY` after imputation. If both the linear term `Y` and the the quadratic term `YY` are variables in the data, then first impute `Y` by calling `mice.impute.quadratic()` on `Y`, and then impute `YY` by passive imputation as `meth["YY"] <- "~I(Y^2)"`. See example section for details. Generally, we would like `YY` to be present in the data if we need to preserve quadratic relations between `YY` and any third variables in the multivariate incomplete data that we might wish to impute.

Author(s)

Gerko Vink (University of Utrecht), <g.vink@uu.nl>

References

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Vink, G., Frank, L.E., van Buuren, S. (2012). Multiple Imputation of Squares. *Sociological Methods & Research*, accepted for publication.

See Also

`mice.impute.pmm`

Examples

```
require(lattice)

# Create Data
B1=.5
B2=.5
X<-rnorm(1000)
XX<-X^2
e<-rnorm(1000, 0, 1)
Y <- B1*X+B2*XX+e
dat <- data.frame(x=X, xx=XX, y=Y)

# Impose 25 percent MCAR Missingness
dat[0 == rbinom(1000, 1, 1-.25), 1:2] <- NA

# Prepare data for imputation
ini <- mice(dat, maxit=0)
meth <- c("quadratic", "~I(x^2)", "")
pred <- ini$pred
pred[,"xx"] <- 0

# Impute data
imp <- mice(dat, meth=meth, pred=pred)

# Pool results
pool(with(imp, lm(y~x+xx)))
```

```
# Plot results
stripplot(imp)
plot(dat$x, dat$xx, col=mdc(1), xlab="x", ylab="xx")
points(complete(imp,1)$x[is.na(dat$x)], complete(imp,1)$xx[is.na(dat$x)], col=mdc(2))
```

`mice.impute.rf` *Imputation by random forests*

Description

Imputes univariate missing data using random forests.

Usage

```
mice.impute.rf(y, ry, x, ntree = 10, ...)
```

Arguments

`y`	Numeric vector with incomplete data
`ry`	Response pattern of `y` (`TRUE` = observed, `FALSE` = missing)
`x`	Design matrix with `length(y)` rows and `p` columns containing complete covariates.
`ntree`	The number of trees to grow. The default is 10.
`...`	Other named arguments passed down to `randomForest()` and `randomForest:::randomForest.defaul`

Details

Imputation of `y` by random forests. The method calls `randomForrest()` which implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression. See Appendix A.1 of Doove et al. (2014) for the definition of the algorithm used. An alternative implementation was independently developed by Shah et al (2014), and is available in the package `CALIBERrfimpute`. Simulations by Shah (Feb 13, 2014) suggested that the quality of the imputation for 10 and 100 trees was identical, so mice 2.22 changed the default number of trees from `ntree = 100` to `ntree = 10`.

Value

Numeric vector of length `sum(!ry)` with imputations

Author(s)

Lisa Doove, Stef van Buuren, Elise Dusseldorp, 2012

References

Doove, L.L., van Buuren, S., Dusseldorp, E. (2014), Recursive partitioning for missing data imputation in the presence of interaction Effects. Computational Statistics \& Data Analysis, 72, 92-104.

Shah, A.D., Bartlett, J.W., Carpenter, J., Nicholas, O., Hemingway, H. (2014), Comparison of random forest and parametric imputation models for imputing missing data using MICE: A CALIBER study. American Journal of Epidemiology, doi: 10.1093/aje/kwt312.

Van Buuren, S.(2012), Flexible imputation of missing data, Boca Raton, FL: Chapman & Hall/CRC.

See Also

mice, mice.impute.cart, randomForest, mice.impute.rfcat, mice.impute.rfcont

Examples

```
library("lattice")

imp <- mice(nhanes2, meth = "rf", ntree = 3)
plot(imp)
```

---

mice.impute.ri *Imputation by the random indicator method for nonignorable data*

Description

Imputes univariate missing data using the random indicator method. This method estimates an offset between the distribution of the observed and missing data using an algorithm that iterates over the response model and the imputation model.

Usage

```
mice.impute.ri(y, ry, x, ri.maxit = 10, ...)
```

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.
ri.maxit	Number of inner iterations
...	Other named arguments passed down to .norm.draw()

Value

A vector of length nmis with imputations.

Author(s)

Shahab Jolani (University of Utrecht) <s.jolani@uu.nl>

References

Jolani, S. (2012). *Dual Imputation Strategies for Analyzing Incomplete Data*. Disseration. University of Utrecht, Dec 7 2012. http://igitur-archive.library.uu.nl/dissertations/2012-1120-200602/Jolani.pdf

---

mice.impute.sample *Imputation by simple random sampling*

Description

Imputes a random sample from the observed y data

Usage

```
mice.impute.sample(y, ry, x = 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.
...	Other named arguments.

Details

This function takes a simple random sample from the observed values in y, and returns these as imputations.

Value

A vector of length nmis with imputations.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

`mice.mids` *Multivariate Imputation by Chained Equations (Iteration Step)*

Description

Takes a `mids` object, and produces a new object of class `mids`.

Usage

```
mice.mids(obj, maxit = 1, diagnostics = TRUE, printFlag = TRUE, ...)
```

Arguments

`obj`	An object of class `mids`, typically produces by a previous call to `mice()` or `mice.mids()`
`maxit`	The number of additional Gibbs sampling iterations.
`diagnostics`	A Boolean flag. If `TRUE`, diagnostic information will be appended to the value of the function. If `FALSE`, only the imputed data are saved. The default is `TRUE`.
`printFlag`	A Boolean flag. If `TRUE`, diagnostic information during the Gibbs sampling iterations will be written to the command window. The default is `TRUE`.
`...`	Named arguments that are passed down to the elementary imputation functions.

Details

This function enables the user to split up the computations of the Gibbs sampler into smaller parts. This is useful for the following reasons:

- RAM memory may become easily exhausted if the number of iterations is large. Returning to prompt/session level may alleviate these problems.
- The user can compute customized convergence statistics at specific points, e.g. after each iteration, for monitoring convergence. - For computing a 'few extra iterations'.

Note: The imputation model itself is specified in the `mice()` function and cannot be changed with `mice.mids`. The state of the random generator is saved with the `mids` object.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Van Buuren, S., Groothuis-Oudshoorn, K. (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

`complete`, `mice`, `set.seed`, `mids`

Examples

```
imp1 <- mice(nhanes,maxit=1)
imp2 <- mice.mids(imp1)

# yields the same result as
imp <- mice(nhanes,maxit=2)

# for example:
#
# > imp$imp$bmi[1,]
#      1    2    3    4    5
# 1 30.1 35.3 33.2 35.3 27.5
# > imp2$imp$bmi[1,]
#      1    2    3    4    5
# 1 30.1 35.3 33.2 35.3 27.5
#
```

`mice.theme` *Set the theme for the plotting Trellis functions*

Description

The `mice.theme()` function sets default choices for Trellis plots that are built into **mice**.

Usage

```
mice.theme(transparent = TRUE, alpha.fill = 0.3)
```

Arguments

`transparent`	A logical indicating whether alpha-transparancy is allowed. The default is `TRUE`.
`alpha.fill`	A numerical values between 0 and 1 that indicates the default alpha value for fills.

Value

`mice.theme()` returns a named list that can be used as a theme in the functions in **lattice**. By default, the `mice.theme()` function sets `transparent <- TRUE` if the current device `.Device` supports semi-transparent colors.

Author(s)

Stef van Buuren 2011

mids-class *Multiply imputed data set (*mids*)*

Description

The mids object contains a multiply imputed data set. The mids object is generated by the mice() and mice.mids() functions. The mids class of objects has methods for the following generic functions: print, summary, plot.

Slots

.Data: Object of class "list" containing the following slots:

call: The call that created the object.

data: A copy of the incomplete data set.

m: The number of imputations.

nmis: An array containing the number of missing observations per column.

imp: A list of ncol(data) components with the generated multiple imputations. Each part of the list is a nmis[j] by m matrix of imputed values for variable j.

method: A vector of strings of length(ncol(data)) specifying the elementary imputation method per column.

predictorMatrix: A square matrix of size ncol(data) containing integers specifying the predictor set.

visitSequence: The sequence in which columns are visited.

post: A vector of strings of length ncol(data) with commands for post-processing

seed: The seed value of the solution.

iteration: Last Gibbs sampling iteration number.

lastSeedValue: The most recent seed value.

chainMean: A list of m components. Each component is a length(visitSequence) by maxit matrix containing the mean of the generated multiple imputations. The array can be used for monitoring convergence. Note that observed data are not present in this mean.

chainVar: A list with similar structure of chainMean, containing the covariances of the imputed values.

loggedEvents: A data.frame with six columns containing warnings, corrective actions, and other inside info.

pad: A list containing various settings of the padded imputation model, i.e. the imputation model after creating dummy variables. Normally, this list is only useful for error checking. List members are pad$data (data padded with columns for factors), pad$predictorMatrix (predictor matrix for the padded data), pad$method (imputation methods applied to the padded data), the vector pad$visitSequence (the visit sequence applied to the padded data), pad$post (post-processing commands for padded data) and categories (a matrix containing descriptive information about the padding operation).

`loggedEvents`: A matrix with six columns containing a record of automatic removal actions. It is `NULL` is no action was made. At initialization the program does the following three actions: 1. A variable that contains missing values, that is not imputed and that is used as a predictor is removed, 2. a constant variable is removed, and 3. a collinear variable is removed. During iteration, the program does the following actions: 1. one or more variables that are linearly dependent are removed (for categorical data, a 'variable' corresponds to a dummy variable), and 2. proportional odds regression imputation that does not converge and is replaced by `polyreg`. Column `it` is the iteration number at which the record was added, `im` is the imputation number, `co` is the column number in the data, `dep` is the name of the name of the dependent variable, `meth` is the imputation method used, and `out` is a (possibly long) character vector with the names of the altered or removed predictors.

Note

Many of the functions of the `mice` package do not use the S4 class definitions, and instead rely on the S3 list equivalent `oldClass(obj) <- "mids"`.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

van Buuren S and Groothuis-Oudshoorn K (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

`mice`, `mira`, `mipo`

`mids2mplus` *Export* `mids` *object to Mplus*

Description

Converts a `mids` object into a format recognized by Mplus, and writes the data and the Mplus input files

Usage

```
mids2mplus(imp, file.prefix = "imp", path = getwd(), sep = "\t",
  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.

Details

This function automates most of the work needed to export a mids object to Mplus. The function writes the multiple imputation datasets, the file that contains the names of the multiple imputation data sets and an Mplus input file. The Mplus input file has the proper file names, so in principle it should run and read the data without alteration. Mplus will recognize the data set as a multiply imputed data set, and do automatic pooling in procedures where that is supported.

Value

The return value is NULL.

Author(s)

Gerko Vink, 2011.

See Also

mids, mids2spss

---

mids2spss *Export* mids *object to SPSS*

Description

Converts a mids object into a format recognized by SPSS, and writes the data and the SPSS syntax files.

Usage

```
mids2spss(imp, filedat = "midsdata.txt", filesps = "readmids.sps",
  path = getwd(), sep = "\t", dec = ".", silent = FALSE)
```

Arguments

imp	The imp argument is an object of class mids, typically produced by the mice() function.
filedat	A character string describing the name of the output data file.
filesps	A character string describing the name of the output syntax file.
path	A character string containing the path of the output file. The value in path is appended to filedat and filesps. By default, files are written to the current R working directory. If path=NULL then no file path appending is done.
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.

Details

This function automates most of the work needed to export a mids object to SPSS. It uses a modified version of writeForeignSPSS() from the foreign package. The modified version allows for a choice of the field and decimal separators, and makes some improvements to the formatting, so that the generated syntax file is amenable to the INCLUDE statement in SPSS.

Below are some things to pay attention to.

The SPSS syntax file has the proper file names and separators set, so in principle it should run and read the data without alteration. SPSS is more strict than R with respect to the paths. Always use the full path, otherwise SPSS may not be able to find the data file.

Factors in R translate into categorical variables in SPSS. The internal coding of factor levels used in R is exported. This is generally acceptable for SPSS. However, when the data are to be combined with existing SPSS data, watch out for any changes in the factor levels codes. The read.spss() in package foreign for reading .sav uses its own internal numbering scheme 1,2,3,... for the levels of a factor. Consequently, changes in factor code can cause discrepancies in factor level when re-imported to SPSS. The solution is to manually recode the factor level in SPSS.

SPSS will recognize the data set as a multiply imputed data set, and do automatic pooling in procedures where that is supported. Note however that pooling is an extra option only available to those who licence the MISSING VALUES module. Without this licence, SPSS will still recognize the structure of the data, but not do any pooling.

Value

The return value is NULL.

Author(s)

Stef van Buuren, dec 2010.

See Also

mids

mipo-class *Multiply imputed pooled analysis (*mipo*)*

Description

The mipo object is generated by the pool function from a link[=mira-class]{mira} object. The mipo class of objects has methods for the following generic functions: print, summary.

Slots

.Data: Object of class "list" containing the following slots:

call: The call that created the mipo object.

call1: The call that created the mira object that was used in call.

call2: The call that created the mids object that was used in call1.

data: A copy of the incomplete data set.

nmis: An array containing the number of missing observations per column.

m: Number of multiple imputations.

qhat: An m by npar matrix containing the complete data estimates for the npar parameters of the m complete data analyses.

u: An m by npar by npar array containing the variance-covariance matrices of the estimates of the m complete data analyses.

qbar: The average of complete data estimates. The multiple imputation estimate.

ubar: The average of the variance-covariance matrix of the complete data estimates.

b: The between imputation variance-covariance matrix for the estimates.

t: The total variance-covariance matrix for the estimates.

r: Relative increases in variance due to missing data.

dfcom: Degrees of freedom in the hypothetically complete data: the sample size minus the number of free parameters.

df: Degrees of freedom associated with the t-statistics.

fmi: Fraction of missing information.

lambda: Proportion of the variation attributable to the missing data: (b+b/m)/t.

Note

The functions of the mice package do not use the S4 class definitions, and instead rely on the S3 list equivalent oldClass(obj) <- "mipo".

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

pool, mids, mira

---

mira-class *Multiply imputed repeated analyses (*mira*)*

Description

The mira object is generated by the with.mids() function. The as.mira() function takes the results of repeated complete-data analysis stored as a list, and turns it into a mira object that can be pooled. Pooling requires that coef() and vcov() methods are available for fitted object. The mira class of objects has methods for the following generic functions: print, summary.

Slots

#'

Object of class "list" containing the following slots:

.Data: The call that created the object.

call: The call that created the mids object that was used in call.

call1: The call that created the mids object that was used in call.

nmis: An array containing the number of missing observations per column.

analyses: A list of m components containing the individual fit objects from each of the m complete data analyses.

Note

Many of the functions of the mice package do not use the S4 class definitions, and instead rely on the S3 list equivalent oldClass(obj) <- "mira".

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

with.mids, mids, mipo

`nelsonaalen` *Cumulative hazard rate or Nelson-Aalen estimator*

Description

Calculates the cumulative hazard rate (Nelson-Aalen estimator)

Usage

```
nelsonaalen(data, timevar, statusvar)
```

Arguments

`data`	A data frame containing the data.
`timevar`	The name of the time variable in `data`.
`statusvar`	The name of the event variable, e.g. death in `data`.

Details

This function is useful for imputing variables that depend on survival time. White and Royston (2009) suggested using the cumulative hazard to the survival time H0(T) rather than T or log(T) as a predictor in imputation models. See section 7.1 of Van Buuren (2012) for an example.

Value

A vector with `nrow(data)` elements containing the Nelson-Aalen estimates of the cumulative hazard function.

Author(s)

Stef van Buuren, 2012

References

White, I. R., Royston, P. (2009). Imputing missing covariate values for the Cox model. *Statistics in Medicine*, *28*(15), 1982-1998.

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman \& Hall/CRC Press.

Examples

```
require(MASS)

leuk$status <- 1  ## no censoring occurs in leuk data (MASS)
ch <- nelsonaalen(leuk, time, status)
plot(x = leuk$time, y = ch, ylab='Cumulative hazard', xlab='Time')

### See example on http://www.engineeredsoftware.com/lmar/pe_cum_hazard_function.htm
```

```
time <- c(43, 67, 92, 94, 149, rep(149,7))
status <- c(rep(1,5),rep(0,7))
eng <- data.frame(time, status)
ch <- nelsonaalen(eng, time, status)
plot(x = time, y = ch, ylab='Cumulative hazard', xlab='Time')
```

---

nhanes *NHANES example - all variables numerical*

Description

A small data set with non-monotone missing values.

Format

A data frame with 25 observations on the following 4 variables.

age Age group (1=20-39, 2=40-59, 3=60+)

bmi Body mass index (kg/m**2)

hyp Hypertensive (1=no,2=yes)

chl Total serum cholesterol (mg/dL)

Details

A small data set with all numerical variables. The data set nhanes2 is the same data set, but with age and hyp treated as factors.

Source

Schafer, J.L. (1997). *Analysis of Incomplete Multivariate Data.* London: Chapman & Hall. Table 6.14.

See Also

nhanes2

Examples

```
imp <- mice(nhanes)     # create 5 imputed data sets
complete(imp)           # print the first imputed data set
```

nhanes2 *NHANES example - mixed numerical and discrete variables*

Description

A small data set with non-monotone missing values.

Format

A data frame with 25 observations on the following 4 variables.

age Age group (1=20-39, 2=40-59, 3=60+)

bmi Body mass index (kg/m**2)

hyp Hypertensive (1=no,2=yes)

chl Total serum cholesterol (mg/dL)

Details

A small data set with missing data and mixed numerical and discrete variables. The data set nhanes is the same data set, but with all data treated as numerical.

Source

Schafer, J.L. (1997). *Analysis of Incomplete Multivariate Data.* London: Chapman & Hall. Table 6.14.

See Also

nhanes

Examples

```
imp <- mice(nhanes2)     # create 5 imputed data sets
complete(imp)           # print the first imputed data set
```

norm.draw *Draws values of beta and sigma by Bayesian linear regression*

Description

This function draws random values of beta and sigma under the Bayesian linear regression model as described in Rubin (1987, p. 167). This function can be called by user-specified imputation functions.

Usage

```
norm.draw(y, ry, x, ridge = 1e-05, ...)

.norm.draw(y, ry, x, ridge = 1e-05, ...)
```

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	A small numerical value specifying the size of the ridge used. The default value ridge = 1e-05 represents a compromise between stability and unbiasedness. Decrease ridge if the data contain many junk variables. Increase ridge for highly collinear data.
...	Other named arguments.

Value

A list containing components coef (least squares estimate), beta (drawn regression weights) and sigma (drawn value of the residual standard deviation).

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

References

Rubin, D.B. (1987). *Multiple imputation for nonresponse in surveys*. New York: Wiley.

---

pattern *Datasets with various missing data patterns*

Description

Four simple datasets with various missing data patterns

Format

list("pattern1") Data with a univariate missing data pattern

list("pattern2") Data with a monotone missing data pattern

list("pattern3") Data with a file matching missing data pattern

list("pattern4") Data with a general missing data pattern

Details

Van Buuren (2012) uses these four artificial datasets to illustrate various missing data patterns.

Source

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Examples

```
require(lattice)
require(MASS)

pattern4

data <- rbind(pattern1, pattern2, pattern3, pattern4)
mdpat <- cbind(expand.grid(rec = 8:1, pat = 1:4, var = 1:3), r=as.numeric(as.vector(is.na(data))))

types <-  c("Univariate","Monotone","File matching","General")
tp41 <- levelplot(r~var+rec|as.factor(pat), data=mdpat,
         as.table=TRUE, aspect="iso",
         shrink=c(0.9),
         col.regions = mdc(1:2),
         colorkey=FALSE,
         scales=list(draw=FALSE),
         xlab="", ylab="",
         between = list(x=1,y=0),
         strip = strip.custom(bg = "grey95", style = 1,
           factor.levels = types))
print(tp41)

md.pattern(pattern4)
p <- md.pairs(pattern4)
p

### proportion of usable cases
p$mr/(p$mr+p$mm)

### outbound statistics
p$rm/(p$rm+p$rr)


fluxplot(pattern2)
```

`plot.mids` *Plot the trace lines of the MICE algorithm*

Description

Trace line plots, also called stream plots or history plots, portray the value of an estimate against the iteration number. The estimate can be anything that you can calculate, but typically are chosen as parameter of scientific interest. The `plot` method for a `mids` object plots the mean and standard deviation of the imputed (not observed) values against the iteration number for each of the m

replications. By default, the function plot the development of the mean and standard deviation for each incomplete variable. On convergence, the streams should intermingle and be free of any trend.

Usage

```
## S3 method for class 'mids'
plot(x, y = NULL, theme = mice.theme(), layout = c(2, 3),
  type = "l", col = 1:10, lty = 1, ...)
```

Arguments

x	An object of class mids
y	A formula that specifies which variables, stream and iterations are plotted. If omitted, all streams, variables and iterations are plotted.
theme	The trellis theme to applied to the graphs. The default is mice.theme().
layout	A vector of length 2 given the number of columns and rows in the plot. The default is c(2, 3).
type	Parameter type of panel.xyplot.
col	Parameter col of panel.xyplot.
lty	Parameter lty of panel.xyplot.
...	Extra arguments for xyplot.

Value

An object of class "trellis".

Author(s)

Stef van Buuren 2011

See Also

mice, mids, xyplot

---

pool *Multiple imputation pooling*

Description

Pools the results of m repeated complete data analysis

Usage

```
pool(object, method = "smallsample")
```

Arguments

object	An object of class mira, produced by with.mids() or as.mira()
method	A string describing the method to compute the degrees of freedom. The default value is "smallsample", which specifies the is Barnard-Rubin adjusted degrees of freedom (Barnard and Rubin, 1999) for small samples. Specifying a different string produces the conventional degrees of freedom as in Rubin (1987).

Details

The function averages the estimates of the complete data model, computes the total variance over the repeated analyses, and computes the relative increase in variance due to nonresponse and the fraction of missing information. The function relies on the availability of

1. the estimates of the model, typically present as 'coefficients' in the fit object
2. an appropriate estimate of the variance-covariance matrix of the estimates per analyses (estimated by vcov.

The function pools also estimates obtained with lme() and lmer(), BUT only the fixed part of the model.

Value

An object of class mipo, which stands for 'multiple imputation pooled outcome'.

Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2009

References

Barnard, J. and Rubin, D.B. (1999). Small sample degrees of freedom with multiple imputation. *Biometrika*, 86, 948-955.

Rubin, D.B. (1987). *Multiple Imputation for Nonresponse in Surveys*. New York: John Wiley and Sons.

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

Pinheiro, J.C. and Bates, D.M. (2000). *Mixed-Effects Models in S and S-PLUS*. Berlin: Springer.

See Also

with.mids, as.mira, vcov

Examples

```
# which vcov methods can R find
methods(vcov)

#
imp <- mice(nhanes)
fit <- with(data=imp,exp=lm(bmi~hyp+chl))
pool(fit)

#Call: pool(object = fit)
#
#Pooled coefficients:
#(Intercept)         hyp         chl
#  22.01313    -1.45578     0.03459
#
#Fraction of information about the coefficients missing due to nonresponse:
#(Intercept)         hyp         chl
#    0.29571     0.05639     0.38759
#> summary(pool(fit))
#                 est      se       t     df Pr(>|t|)    lo 95    hi 95 missing
#(Intercept) 22.01313 4.94086  4.4553 12.016 0.000783 11.24954 32.77673      NA
#hyp         -1.45578 2.26789 -0.6419 20.613 0.528006 -6.17752  3.26596       8
#chl          0.03459 0.02829  1.2228  9.347 0.251332 -0.02904  0.09822      10
#               fmi
#(Intercept) 0.29571
#hyp         0.05639
#chl         0.38759
#
```

`pool.compare` *Compare two nested models fitted to imputed data*

Description

Compares two nested models after m repeated complete data analysis

Usage

```
pool.compare(fit1, fit0, data = NULL, method = "Wald")
```

Arguments

`fit1`	An object of class 'mira', produced by `with.mids()`.
`fit0`	An object of class 'mira', produced by `with.mids()`. The model in `fit0` should be a submodel of `fit1`. Moreover, the variables of the submodel should be the first variables of the larger model and in the same order as in the submodel.
`data`	In case of method 'likelihood' it is necessary to pass also the original `mids` object to the `data` argument. Default value is `NULL`, in case of method='Wald'.

method — A string describing the method to compare the two models. Two kind of comparisons are included so far: 'Wald' and 'likelihood'.

Details

The function is based on the article of Meng and Rubin (1992). The Wald-method can be found in paragraph 2.2 and the likelihoodmethod can be found in paragraph 3. One could use the Wald method for comparison of linear models obtained with e.g. `lm` (in `with.mids()`). The likelihood method should be used in case of logistic regression models obtaind with `glm()` in `with.mids()`. It is assumed that fit1 contains the larger model and the model in `fit0` is fully contained in `fit1`. In case of `method='Wald'`, the null hypothesis is tested that the extra parameters are all zero.

Value

A list containing several components. Component `call` is that call to the `pool.compare` function. Component `call11` is the call that created `fit1`. Component `call12` is the call that created the imputations. Component `call01` is the call that created `fit0`. Compenent `call02` is the call that created the imputations. Components `method` is the method used to compare two models: 'Wald' or 'likelihood'. Component `nmis` is the number of missing entries for each variable. Component `m` is the number of imputations. Component `qhat1` is a matrix, containing the estimated coeffients of the *m* repeated complete data analyses from `fit1`. Component `qhat0` is a matrix, containing the estimated coeffients of the *m* repeated complete data analyses from `fit0`. Component `ubar1` is the mean of the variances of `fit1`, formula (3.1.3), Rubin (1987). Component `ubar0` is the mean of the variances of `fit0`, formula (3.1.3), Rubin (1987). Component `qbar1` is the pooled estimate of `fit1`, formula (3.1.2) Rubin (1987). Component `qbar0` is the pooled estimate of `fit0`, formula (3.1.2) Rubin (1987). Component `Dm` is the test statistic. Component `rm` is the relative increase in variance due to nonresponse, formula (3.1.7), Rubin (1987). Component `df1`: df1 = under the null hypothesis it is assumed that `Dm` has an F distribution with (df1,df2) degrees of freedom. Component `df2`: df2. Component `pvalue` is the P-value of testing whether the larger model is statistically different from the smaller submodel.

Author(s)

Karin Groothuis-Oudshoorn and Stef van Buuren, 2009

References

Li, K.H., Meng, X.L., Raghunathan, T.E. and Rubin, D. B. (1991). Significance levels from repeated p-values with multiply-imputed data. Statistica Sinica, 1, 65-92.

Meng, X.L. and Rubin, D.B. (1992). Performing likelihood ratio tests with multiple-imputed data sets. Biometrika, 79, 103-111.

van Buuren S and Groothuis-Oudshoorn K (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

`lm.mids`, `glm.mids`, `vcov`,

Examples

```
### To compare two linear models:
imp <- mice(nhanes2)
mi1 <- with(data=imp, expr=lm(bmi~age+hyp+chl))
mi0 <- with(data=imp, expr=lm(bmi~age+hyp))
pc  <- pool.compare(mi1, mi0, method='Wald')
pc$spvalue
#            [,1]
#[1,] 0.000293631
#

### Comparison of two general linear models (logistic regression).
## Not run:
imp  <- mice(boys, maxit=2)
fit0 <- with(imp, glm(gen>levels(gen)[1] ~ hgt+hc,family=binomial))
fit1 <- with(imp, glm(gen>levels(gen)[1] ~ hgt+hc+reg,family=binomial))
pool.compare(fit1, fit0, method='likelihood', data=imp)
## End(Not run)
```

`pool.r.squared` *Pooling: R squared*

Description

Pools R^2 of m repeated complete data models.

Usage

```
pool.r.squared(object, adjusted = FALSE)
```

Arguments

`object`	An object of class 'mira', produced by `lm.mids` or `with.mids` with `lm` as modelling function.
`adjusted`	A logical value. If adjusted=TRUE then the adjusted R^2 is calculated. The default value is FALSE.

Details

The function pools the coefficients of determination R^2 or the adjusted coefficients of determination (R^2_a) obtained with the `lm` modelling function. For pooling it uses the Fisher z-transformation.

Value

Returns a 1x4 table with components. Component `est` is the pooled R^2 estimate. Component `lo95` is the 95 % lower bound of the pooled R^2. Component `hi95` is the 95 % upper bound of the pooled R^2. Component `fmi` is the fraction of missing information due to nonresponse.

Author(s)

Karin Groothuis-Oudshoorn and Stef van Buuren, 2009

References

Harel, O (2009). The estimation of R^2 and adjusted R^2 in incomplete data sets using multiple imputation, Journal of Applied Statistics, 36:1109-1118.

Rubin, D.B. (1987). Multiple Imputation for Nonresponse in Surveys. New York: John Wiley and Sons.

van Buuren S and Groothuis-Oudshoorn K (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

`pool`,`pool.scalar`

Examples

```
imp<-mice(nhanes)

fit<-lm.mids(chl~age+hyp+bmi,imp)
pool.r.squared(fit)
pool.r.squared(fit,adjusted=TRUE)

#fit<-lm.mids(chl~age+hyp+bmi,imp)
#
#> pool.r.squared(fit)
#          est     lo 95     hi 95       fmi
#R^2 0.5108041 0.1479687 0.7791927 0.3024413
#
#> pool.r.squared(fit,adjusted=TRUE)
#          est      lo 95    hi 95       fmi
#adj R^2 0.4398066 0.08251427 0.743172 0.3404165
#
```

`pool.scalar` *Multiple imputation pooling: univariate version*

Description

Pools univariate estimates of m repeated complete data analysis

Usage

```
pool.scalar(Q, U, n = 99999, k = 1, method = "smallsample")
```

Arguments

Q	A vector of univariate estimates of m repeated complete data analyses.
U	A vector containing the corresponding m variances of the univariate estimates.
n	A number providing the sample size. If nothing is specified, a large sample n = 99999 is assumed.
k	A number indicating the number of parameters to be estimated. By default, k = 1 is assumed.
method	A string indicatint the method to calculate the degrees of freedom. If method = "smallsample" (the default) then the Barnard-Rubin adjustment for small degrees of freedom is used. Otherwise, the method from Rubin (1987) is used.

Details

The function averages the univariate estimates of the complete data model, computes the total variance over the repeated analyses, and computes the relative increase in variance due to nonresponse and the fraction of missing information.

Value

Returns a list with components. Component m is the number of imputations. Component qhat contains the m univariate estimates of repeated complete data analyses. Component u contains the corresponding m variances of the univariate estimates. Component qbar is the pooled univariate estimate, formula (3.1.2) Rubin (1987). Component ubar is the mean of the variances (i.e. the pooled within-imputation variance), formula (3.1.3) Rubin (1987). Component b is the between-imputation variance, formula (3.1.4) Rubin (1987). Component t is the total variance of the pooled estimated, formula (3.1.5) Rubin (1987). Component r is the relative increase in variance due to nonresponse, formula (3.1.7) Rubin (1987). Component df is the degrees of freedom for t reference distribution, formula (3.1.6) Rubin (1987) or method of Barnard-Rubin (1999) (if method = "smallsample"). Component fmi is the fraction missing information due to nonresponse, formula (3.1.10) Rubin (1987). Component lambda is the proportion of variation due to nonresponse, formula (2.24) Van Buuren (2012).

Author(s)

Karin Groothuis-Oudshoorn and Stef van Buuren, 2009

References

Rubin, D.B. (1987). Multiple Imputation for Nonresponse in Surveys. New York: John Wiley and Sons.

See Also

pool

Examples

```
imp <- mice(nhanes)
m <- imp$m
Q <- rep(NA, m)
U <- rep(NA, m)
for (i in 1:m) {
   Q[i] <- mean(complete(imp, i)$bmi)
   U[i] <- var(complete(imp, i)$bmi) / nrow(nhanes)  # (standard error of estimate)^2
}
pool.scalar(Q, U, method = "rubin")   # Rubin 1987
pool.scalar(Q, U, n = nrow(nhanes), k = 1)  # Barnard-Rubin 1999
```

---

popmis *Hox pupil popularity data with missing popularity scores*

Description

Hox pupil popularity data with some missing popularity scores

Format

A data frame with 2000 rows and 7 columns:

pupil Pupil number within school

school School number

popular Pupil popularity with 848 missing entries

sex Pupil gender

texp Teacher experience (years)

const Constant intercept term

teachpop Teacher popularity

Details

The original, complete dataset was generated by Joop Hox as an example of well-behaved multilevel data set. The distributed data contains missing data in pupil popularity.

Source

Hox, J. J. (2002) *Multilevel analysis. Techniques and applications.* Mahwah, NJ: Lawrence Erlbaum.

Examples

```
popmis[1:3,]
```

pops *Project on preterm and small for gestational age infants (POPS)*

Description

Subset of data from the POPS study, a national, prospective study on preterm children, including all liveborn infants <32 weeks gestional age and/or <1500 g from 1983 (n = 1338).

Format

pops is a data frame with 959 rows and 86 columns. pops.pred is the 86 by 86 binary predictor matrix used for specifying the multiple imputation model.

Details

The data set concerns of subset of 959 children that survived up to the age of 19 years.

Hille et al (2005) divided the 959 survivors into three groups: Full responders (examined at an outpatient clinic and completed the questionnaires, n = 596), postal responders (only completed the mailed questionnaires, n = 109), non-responders (did not respond to any of the mailed requests or telephone calls, or could not be traced, n = 254).

Compared to the postal and non-responders, the full response group consists of more girls, contains more Dutch children, has higher educational and social economic levels and has fewer handicaps. The responders form a highly selective subgroup in the total cohort.

Multiple imputation of this data set has been described in Hille et al (2007) and Van Buuren (2012), chapter 8.

Source

Hille, E. T. M., Elbertse, L., Bennebroek Gravenhorst, J., Brand, R., Verloove-Vanhorick, S. P. (2005). Nonresponse bias in a follow-up study of 19-year-old adolescents born as preterm infants. Pediatrics, 116(5):662666.

Hille, E. T. M., Weisglas-Kuperus, N., Van Goudoever, J. B., Jacobusse, G. W., Ens-Dokkum, M. H., De Groot, L., Wit, J. M., Geven, W. B., Kok, J. H., De Kleine, M. J. K., Kollee, L. A. A., Mulder, A. L. M., Van Straaten, H. L. M., De Vries, L. S., Van Weissenbruch, M. M., Verloove-Vanhorick, S. P. (2007). Functional outcomes and participation in young adulthood for very preterm and very low birth weight infants: The Dutch project on preterm and small for gestational age infants at 19 years of age. Pediatrics, 120(3):587595.

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Examples

```
pops <- data(pops)
```

potthoffroy *Potthoff-Roy data*

Description

Data from Potthoff-Roy (1964) with repeated measures on dental fissures.

Format

`tbs` is a data frame with 27 rows and 6 columns:

id Person number

sex Sex M/F

d8 Distance at age 8 years

d10 Distance at age 10 years

d12 Distance at age 12 years

d14 Distance at age 14 years

Details

This data set is the famous Potthoff-Roy data, used to demonstrate MANOVA on repeated measure data. Potthoff and Roy (1964) published classic data on a study in 16 boys and 11 girls, who at ages 8, 10, 12, and 14 had the distance (mm) from the center of the pituitary gland to the pteryomaxillary fissure measured. Changes in pituitary-pteryomaxillary distances during growth is important in orthodontic therapy. The goals of the study were to describe the distance in boys and girls as simple functions of age, and then to compare the functions for boys and girls. The data have been reanalyzed by many authors including Jennrich and Schluchter (1986), Little and Rubin (1987), Pinheiro and Bates (2000), Verbeke and Molenberghs (2000) and Molenberghs and Kenward (2007). See Chapter 9 of Van Buuren (2012) for a challengeing exercise using these data.

Source

Potthoff, R. F., Roy, S. N. (1964). A generalized multivariate analysis of variance model usefully especially for growth curve problems. *Biometrika*, *51*(3), 313-326.

Little, R. J. A., Rubin, D. B. (1987). *Statistical Analysis with Missing Data.* New York: John Wiley & Sons.

Van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman \& Hall/CRC Press.

Examples

```
### create missing values at age 10 as in Little and Rubin (1987)

phr <- potthoffroy
idmis <- c(3,6,9,10,13,16,23,24,27)
phr[idmis, 4] <- NA
```

```
phr

md.pattern(phr)
```

`print.mids` *Print a* `mids` *object*

Description

Print a `mids` object

Print a `mira` object

Print a `mipo` object

Usage

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

## S3 method for class 'mira'
print(x, ...)

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

Arguments

`x`	Object of class `mids`, `mira` or `mipo`
`...`	Other parameters passed down to `print.default()`

Value

`NULL`

`NULL`

`NULL`

See Also

`mids`

`mira`

`mipo`

quickpred *Quick selection of predictors from the data*

Description

Selects predictors according to simple statistics

Usage

```
quickpred(data, mincor = 0.1, minpuc = 0, include = "", exclude = "",
  method = "pearson")
```

Arguments

data	Matrix or data frame with incomplete data.
mincor	A scalar, numeric vector (of size ncol(data)) or numeric matrix (square, of size ncol(data) specifiying the minimum threshold(s) against which the absolute correlation in the data is compared.
minpuc	A scalar, vector (of size ncol(data)) or matrix (square, of size ncol(data) specifiying the minimum threshold(s) for the proportion of usable cases.
include	A string or a vector of strings containing one or more variable names from names(data). Variables specified are always included as a predictor.
exclude	A string or a vector of strings containing one or more variable names from names(data). Variables specified are always excluded as a predictor.
method	A string specifying the type of correlation. Use 'pearson' (default), 'kendall' or 'spearman'. Can be abbreviated.

Details

This function creates a predictor matrix using the variable selection procedure described in Van Buuren et al.~(1999, p.~687–688). The function is designed to aid in setting up a good imputation model for data with many variables.

Basic workings: The procedure calculates for each variable pair (i.e. target-predictor pair) two correlations using all available cases per pair. The first correlation uses the values of the target and the predictor directly. The second correlation uses the (binary) response indicator of the target and the values of the predictor. If the largest (in absolute value) of these correlations exceeds mincor, the predictor will be added to the imputation set. The default value for mincor is 0.1.

In addition, the procedure eliminates predictors whose proportion of usable cases fails to meet the minimum specified by minpuc. The default value is 0, so predictors are retained even if they have no usable case.

Finally, the procedure includes any predictors named in the include argument (which is useful for background variables like age and sex) and eliminates any predictor named in the exclude argument. If a variable is listed in both include and exclude arguments, the include argument takes precedence.

Advanced topic: mincor and minpuc are typically specified as scalars, but vectors and squares matrices of appropriate size will also work. Each element of the vector corresponds to a row of the predictor matrix, so the procedure can effectively differentiate between different target variables. Setting a high values for can be useful for auxilary, less important, variables. The set of predictor for those variables can remain relatively small. Using a square matrix extends the idea to the columns, so that one can also apply cellwise thresholds.

Value

A square binary matrix of size ncol(data).

Author(s)

Stef van Buuren, Aug 2009

References

van Buuren, S., Boshuizen, H.C., Knook, D.L. (1999) Multiple imputation of missing blood pressure covariates in survival analysis. *Statistics in Medicine*, **18**, 681–694.

van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mice, mids

Examples

```
# default: include all predictors with absolute correlation over 0.1
quickpred(nhanes)

# all predictors with absolute correlation over 0.4
quickpred(nhanes, mincor=0.4)

# include age and bmi, exclude chl
quickpred(nhanes, mincor=0.4, inc=c('age','bmi'), exc='chl')

# only include predictors with at least 30% usable cases
quickpred(nhanes, minpuc=0.3)

# use low threshold for bmi, and high thresholds for hyp and chl
pred <- quickpred(nhanes, mincor=c(0,0.1,0.5,0.5))
pred

# use it directly from mice
imp <- mice(nhanes, pred=quickpred(nhanes, minpuc=0.25, include='age'))
```

rbind.mids *Rowwise combination of a* mids *object.*

Description

Append mids objects by rows

Usage

```
rbind.mids(x, y, ...)
```

Arguments

x	A mids object.
y	A mids object or a data.frame, matrix, factor or vector.
...	Additional data.frame, matrix, vector or factor. These can be given as named arguments.

Details

This function combines two mids objects rowwise into a single mids object or combines a mids object and a vector, matrix, factor or dataframe rowwise into a mids object. The number of columns in the (incomplete) data x$data and y (or y$data if y is a mids object) should be equal. If y is a mids object then the number of imputations in x and y should be equal.

Value

An S3 object of class mids

Note

Component call is a vector, with first argument the mice() statement that created x and second argument the call to rbind.mids(). Component data is the rowwise combination of the (incomplete) data in x and y. Component m is equal to x$m. Component nmis is an array containing the number of missing observations per column, defined as x$nmis + y$nmis. Component imp is a list of nvar components with the generated multiple imputations. Each part of the list is a nmis[j] by m matrix of imputed values for variable j. If y is a mids object then imp[[j]] equals rbind(x$imp[[j]], y$imp[[j]]); otherwise the original data of y will be copied into this list, including the missing values of y then y is not imputed. Component method is a vector of strings of length(nvar) specifying the elementary imputation method per column defined as x$method. Component predictorMatrix is a square matrix of size ncol(data) containing the predictor set defined as x$predictorMatrix. Component visitSequence is the sequence in which columns are visited, defined as x$visitSequence. Component seed is the seed value of the solution, x$seed. Component iteration is the last Gibbs sampling iteration number, x$iteration. Component lastSeedValue is the most recent seed value, x$lastSeedValue Component chainMean is set to NA. Component chainVar is set to NA. Component pad is set to x$pad, a list containing various settings of the padded imputation model, i.e. the imputation model after creating dummy variables. Component loggedEvents is set to x$loggedEvents.

Author(s)

Karin Groothuis-Oudshoorn, Stef van Buuren, 2009

References

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

cbind.mids, ibind, mids

---

selfreport *Self-reported and measured BMI*

Description

Dataset containing height and weight data (measured, self-reported) from two studies.

Format

A data frame with 2060 rows and 15 variables:

src Study, either krul or mgg (factor)

id Person identification number

pop Population, all NL (factor)

age Age of respondent in years

sex Sex of respondent (factor)

hm Height measured (cm)

wm Weight measured (kg)

hr Height reported (cm)

wr Weight reported (kg)

prg Pregnancy (factor), all Not pregnant

edu Educational level (factor)

etn Ethnicity (factor)

web Obtained through web survey (factor)

bm BMI measured (kg/m2)

br BMI reported (kg/m2)

Details

This dataset combines two datasets: `krul` data (Krul, 2010) (1257 persons) and the `mgg` data (Van Keulen 2011; Van der Klauw 2011) (803 persons). The `krul` dataset contains height and weight (both measures and self-reported) from 1257 Dutch adults, whereas the `mgg` dataset contains self-reported height and weight for 803 Dutch adults. Section 7.3 in Van Buuren (2012) shows how the missing measured data can be imputed in the `mgg` data, so corrected prevalence estimates can be calculated.

Source

Krul, A., Daanen, H. A. M., Choi, H. (2010). Self-reported and measured weight, height and body mass index (BMI) in Italy, The Netherlands and North America. *European Journal of Public Health*, *21*(4), 414-419.

Van Keulen, H.M.,, Chorus, A.M.J., Verheijden, M.W. (2011). *Monitor Convenant Gezond Gewicht Nulmeting (determinanten van) beweeg- en eetgedrag van kinderen (4-11 jaar), jongeren (12-17 jaar) en volwassenen (18+ jaar)*. TNO/LS 2011.016. Leiden: TNO. http://www.gezondeschool.info/object_binary/o11783_2011-016-monitor-convenant-gezond-gewicht-def.pdf

Van der Klauw, M., Van Keulen, H.M., Verheijden, M.W. (2011). *Monitor Convenant Gezond Gewicht Beweeg- en eetgedrag van kinderen (4-11 jaar), jongeren (12-17 jaar) en volwassenen (18+ jaar) in 2010 en 2011.* TNO/LS 2011.055. Leiden: TNO. (in Dutch) http://www.convenantgezondgewicht.nl/download/131/2011_055_monitor_convenant_gezond_gewicht.pdf

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC Press.

Examples

```
md.pattern(selfreport[,c("age","sex","hm","hr","wm","wr")])

### FIMD Section 7.3.5 Application

bmi <- function(h,w){return(w/(h/100)^2)}
init <- mice(selfreport,maxit=0)
meth <- init$meth
meth["bm"] <- "~bmi(hm,wm)"
pred <- init$pred
pred[,c("src","id","web","bm","br")] <- 0
imp <- mice(selfreport, pred=pred, meth=meth, seed=66573, maxit=2, m=1)
## imp <- mice(selfreport, pred=pred, meth=meth, seed=66573, maxit=20, m=10)

### Like FIMD Figure 7.6

cd <- complete(imp, 1)
xy <- xy.coords(cd$bm, cd$br-cd$bm)
plot(xy,col=mdc(2),xlab="Measured BMI",ylab="Reported - Measured BMI",
    xlim=c(17,45),ylim=c(-5,5), type="n",lwd=0.7)
polygon(x=c(30,20,30),y=c(0,10,10),col="grey95",border=NA)
polygon(x=c(30,40,30),y=c(0,-10,-10),col="grey95",border=NA)
abline(0,0,lty=2,lwd=0.7)
```

```
idx <- cd$src=="krul"
xyc <- xy; xyc$x <- xy$x[idx]; xyc$y <- xy$y[idx]
xys <- xy; xys$x <- xy$x[!idx]; xys$y <- xy$y[!idx]
points(xyc,col=mdc(1), cex=0.7)
points(xys,col=mdc(2), cex=0.7)
lines(lowess(xyc),col=mdc(4),lwd=2)
lines(lowess(xys),col=mdc(5),lwd=2)
text(1:4,x=c(40,28,20,32),y=c(4,4,-4,-4),cex=3)
box(lwd=1)
```

`squeeze` *Squeeze the imputed values to be within specified boundaries.*

Description

This function replaces any values in `x` that are lower than `bounds[1]` by `bounds[1]`, and replaces any values higher than `bounds[2]` by `bounds[2]`.

Usage

```
squeeze(x, bounds = c(min(x[r]), max(x[r])), r = rep(TRUE, length(x)))
```

Arguments

`x`	A numerical vector with values
`bounds`	A numerical vector of length 2 containing the lower and upper bounds. By default, the bounds are to the minimum and maximum values in `x`.
`r`	A logical vector of length `length(x)` that is used to select a subset in `x` before calculating automatic bounds.

Value

A vector of length `length(x)`.

Author(s)

Stef van Buuren, 2011.

stripplot.mids *Stripplot of observed and imputed data*

Description

Plotting methods for imputed data using **lattice**. `stripplot` produces one-dimensional scatterplots. The function automatically separates the observed and imputed data. The functions extend the usual features of **lattice**.

Usage

```
## S3 method for class 'mids'
stripplot(x, data, na.groups = NULL, groups = NULL,
  as.table = TRUE, theme = mice.theme(), allow.multiple = TRUE,
  outer = TRUE,
  drop.unused.levels = lattice.getOption("drop.unused.levels"),
  panel = lattice.getOption("panel.stripplot"),
  default.prepanel = lattice.getOption("prepanel.default.stripplot"),
  jitter.data = TRUE, horizontal = FALSE, ..., subscripts = TRUE,
  subset = TRUE)
```

Arguments

`x`	A `mids` object, typically created by `mice()` or `mice.mids()`.
`data`	Formula that selects the data to be plotted. This argument follows the **lattice** rules for *formulas*, describing the primary variables (used for the per-panel display) and the optional conditioning variables (which define the subsets plotted in different panels) to be used in the plot. The formula is evaluated on the complete data set in the `long` form. Legal variable names for the formula include `names(x$data)` plus the two administrative factors `.imp` and `.id`. **Extended formula interface:** The primary variable terms (both the LHS `y` and RHS `x`) may consist of multiple terms separated by a '+' sign, e.g., `y1 + y2 ~ x | a * b`. This formula would be taken to mean that the user wants to plot both `y1 ~ x | a * b` and `y2 ~ x | a * b`, but with the `y1 ~ x` and `y2 ~ x` in *separate panels*. This behavior differs from standard **lattice**. *Only combine terms of the same type*, i.e. only factors or only numerical variables. Mixing numerical and categorical data occasionally produces odds labeling of vertical axis. For convience, in `stripplot()` and `bwplot` the formula `y~.imp` may be abbreviated as `y`. This applies only to a single `y`, and does not (yet) work for `y1+y2~.imp`.
`na.groups`	An expression evaluating to a logical vector indicating which two groups are distinguished (e.g. using different colors) in the display. The environment in which this expression is evaluated in the response indicator `is.na(x$data)`.

The default na.group = NULL constrasts the observed and missing data in the LHS y variable of the display, i.e. groups created by is.na(y). The expression y creates the groups according to is.na(y). The expression y1 & y2 creates groups by is.na(y1) & is.na(y2), and y1 | y2 creates groups as is.na(y1) | is.na(y2), and so on.

groups — This is the usual groups arguments in **lattice**. It differs from na.groups because it evaluates in the completed data data.frame(complete(x, "long", inc=TRUE)) (as usual), whereas na.groups evaluates in the response indicator. See xyplot for more details. When both na.groups and groups are specified, na.groups takes precedence, and groups is ignored.

theme — A named list containing the graphical parameters. The default function mice.theme produces a short list of default colors, line width, and so on. The extensive list may be obtained from trellis.par.get(). Global graphical parameters like col or cex in high-level calls are still honored, so first experiment with the global parameters. Many setting consists of a pair. For example, mice.theme defines two symbol colors. The first is for the observed data, the second for the imputed data. The theme settings only exist during the call, and do not affect the trellis graphical parameters.

jitter.data — See panel.xyplot.

horizontal — See xyplot.

as.table — See xyplot.

panel — See xyplot.

default.prepanel — See xyplot.

outer — See xyplot.

allow.multiple — See xyplot.

drop.unused.levels — See xyplot.

subscripts — See xyplot.

subset — See xyplot.

... — Further arguments, usually not directly processed by the high-level functions documented here, but instead passed on to other functions.

Details

The argument na.groups may be used to specify (combinations of) missingness in any of the variables. The argument groups can be used to specify groups based on the variable values themselves. Only one of both may be active at the same time. When both are specified, na.groups takes precedence over groups.

Use the subset and na.groups together to plots parts of the data. For example, select the first imputed data set by by subset=.imp==1.

Graphical paramaters like col, pch and cex can be specified in the arguments list to alter the plotting symbols. If length(col)==2, the color specification to define the observed and missing groups. col[1] is the color of the 'observed' data, col[2] is the color of the missing or imputed data.

A convenient color choice is `col=mdc(1:2)`, a transparent blue color for the observed data, and a transparent red color for the imputed data. A good choice is `col=mdc(1:2), pch=20, cex=1.5`. These choices can be set for the duration of the session by running `mice.theme()`.

Value

The high-level functions documented here, as well as other high-level Lattice functions, return an object of class `"trellis"`. The `update` method can be used to subsequently update components of the object, and the `print` method (usually called by default) will plot it on an appropriate plotting device.

Note

The first two arguments (`x` and `data`) are reversed compared to the standard Trellis syntax implemented in **lattice**. This reversal was necessary in order to benefit from automatic method dispatch.

In **mice** the argument `x` is always a `mids` object, whereas in **lattice** the argument `x` is always a formula.

In **mice** the argument `data` is always a formula object, whereas in **lattice** the argument `data` is usually a data frame.

All other arguments have identical interpretation.

Author(s)

Stef van Buuren

References

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*, Springer. http://lmdvr.r-forge.r-project.org/

van Buuren S and Groothuis-Oudshoorn K (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

`mice`, `xyplot`, `densityplot`, `bwplot`, `Lattice` for an overview of the package, as well as `stripplot`, `panel.stripplot`, `print.trellis`, `trellis.par.set`

Examples

```
require(lattice)

imp <- mice(boys, maxit=1)

### stripplot, all numerical variables
## Not run: stripplot(imp)

### same, but with improved display
## Not run: stripplot(imp, col=c("grey",mdc(2)),pch=c(1,20))
```

```
### distribution per imputation of height, weight and bmi
### labeled by their own missingness
## Not run: stripplot(imp, hgt+wgt+bmi~.imp, cex=c(2,4), pch=c(1,20),jitter=FALSE,
layout=c(3,1))
## End(Not run)

### same, but labeled with the missingness of wgt (just four cases)
## Not run: stripplot(imp, hgt+wgt+bmi~.imp, na=wgt, cex=c(2,4), pch=c(1,20),jitter=FALSE,
layout=c(3,1))
## End(Not run)

### distribution of age and height, labeled by missingness in height
### most height values are missing for those around
### the age of two years
### some additional missings occur in region WEST
## Not run: stripplot(imp, age+hgt~.imp|reg, hgt, col=c(hcl(0,0,40,0.2), mdc(2)),pch=c(1,20))

### heavily jitted relation between two categorical variables
### labeled by missingness of gen
### aggregated over all imputed data sets
## Not run: stripplot(imp, gen~phb, factor=2, cex=c(8,1), hor=TRUE)

### circle fun
stripplot(imp, gen~.imp, na = wgt, factor = 2, cex = c(8.6),
         hor = FALSE, outer = TRUE, scales = "free", pch = c(1,19))
```

`summary.mira` *Summary of a* `mira` *object*

Description

Summary of a `mira` object

Summary of a `mipo` object

Summary of a `mids` object

Usage

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

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

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

Arguments

object	A mira object
...	Other parameters passed down to print() and summary()

Value

NULL

A table containing summary statistis of the pooled analysis

NULL

See Also

mira

mipo

mids

---

supports.transparent *Supports semi-transparent foreground colors?*

Description

This function is used by mdc() to find out whether the current device supports semi-transparent foreground colors.

Usage

```
supports.transparent()
```

Details

The function calls the function dev.capabilities() from the package grDevices. The function return FALSE if the status of the current device is unknown.

Value

TRUE or FALSE

See Also

mdc dev.capabilities

Examples

```
supports.transparent()
```

tbc *Terneuzen birth cohort*

Description

Data of subset of the Terneuzen Birth Cohort data on child growth.

Format

`tbs` is a data frame with 3951 rows and 11 columns:

id Person number

occ Occasion number

nocc Number of occasions

first Is this the first record for this person? (TRUE/FALSE)

typ Type of data (all observed)

age Age (years)

sex Sex 1=M, 2=F

hgt.z Height Z-score

wgt.z Weight Z-score

bmi.z BMI Z-score

ao Adult overweight (0=no, 1=yes)

`tbc.target` is a data frame with 2612 rows and 3 columns:

id Person number

ao Adult overweight (0=no, 1=yes)

bmi.z.jv BMI Z-score as young adult (18-29 years)

Details

This `tbc` data set is a random subset of persons from a much larger collection of data from the Terneuzen Birth Cohort. The total cohort comprises of 2604 unique persons, whereas the subset in `tbc` covers 306 persons. The `tbc.target` is an auxiliary data set containing two outcomes at adult age. For more details, see De Kroon et al (2008, 2010, 2011). The imputation methodology is explained in Chapter 9 of Van Buuren (2012).

Source

De Kroon, M. L. A., Renders, C. M., Kuipers, E. C., van Wouwe, J. P., van Buuren, S., de Jonge, G. A., Hirasing, R. A. (2008). Identifying metabolic syndrome without blood tests in young adults - The Terneuzen birth cohort. *European Journal of Public Health*, *18*(6), 656-660.

De Kroon, M. L. A., Renders, C. M., Van Wouwe, J. P., Van Buuren, S., Hirasing, R. A. (2010). The Terneuzen birth cohort: BMI changes between 2 and 6 years correlate strongest with adult overweight. *PLoS ONE*, *5*(2), e9155.

De Kroon, M. L. A. (2011). *The Terneuzen Birth Cohort. Detection and Prevention of Overweight and Cardiometabolic Risk from Infancy Onward.* Disseration, Vrije Universiteit, Amsterdam. http://dare.ubvu.vu.nl/handle/1871/23806

Van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman \& Hall/CRC Press.

Examples

```
data <- tbc
md.pattern(data)
```

`version` *Echoes the package version number*

Description

Echoes the package version number

Usage

```
version(pkg = "mice")
```

Arguments

`pkg`	A character vector with the package name.

Value

A character vector containing the package name, version number and installed directory.

Author(s)

Stef van Buuren, Oct 2010

Examples

```
version()
version("base")
```

walking *Walking disability data*

Description

Two items YA and YB measuring walking disability in samples A, B and E.

Format

A data frame with 890 rows on the following 5 variables:

sex Sex of respondent (factor)

age Age of respondent

YA Item administered in samples A and E (factor)

YB Item administered in samples B and E (factor)

src Source: Sample A, B or E (factor)

Details

Example dataset to demonstrate imputation of two items (YA and YB). Item YA is administered to sample A and sample E, item YB is administered to sample B and sample E, so sample E acts as a bridge study. Imputation using a bridge study is better than simple equating or than imputation under independence.

Item YA corresponds to the HAQ8 item, and item YB corresponds to the GAR9 items from Van Buuren et al (2005). Sample E (as well as sample B) is the Euridiss study (n=292), sample A is the ERGOPLUS study (n=306).

See Van Buuren (2012) chapter 7 for more details on the imputation methodology.

References

van Buuren, S., Eyres, S., Tennant, A., Hopman-Rock, M. (2005). Improving comparability of existing data by Response Conversion. *Journal of Official Statistics*, **21**(1), 53-72.

van Buuren, S. (2012). *Flexible Imputation of Missing Data.* Boca Raton, FL: Chapman & Hall/CRC.

Examples

```
md.pattern(walking)

micemill <- function(n) {
 for (i in 1:n) {
   imp <<- mice.mids(imp) # global assignment
   cors <- with(imp, cor(as.numeric(YA),
                         as.numeric(YB),
                         method="kendall"))
   tau <<- rbind(tau, getfit(cors, s=TRUE))  # global assignment
```

```
 }
}

plotit <- function()
 matplot(x=1:nrow(tau),y=tau,
       ylab=expression(paste("Kendall's ",tau)),
         xlab="Iteration", type="l", lwd=1,
       lty=1:10,col="black")

tau <- NULL
imp <- mice(walking, max=0, m=10, seed=92786)
pred <- imp$pred
pred[,c("src","age","sex")] <- 0
imp <- mice(walking, max=0, m=3, seed=92786, pred=pred)
micemill(5)
plotit()

### to get figure 7.8 van Buuren (2012) use m=10 and micemill(20)
```

`windspeed` *Subset of Irish wind speed data*

Description

Subset of Irish wind speed data

Format

A data frame with 433 rows and 6 columns containing the daily average wind speeds within the period 1961-1978 at meteorological stations in the Republic of Ireland. The data are a random sample from a larger data set.

RochePt Roche Point

Rosslare Rosslare

Shannon Shannon

Dublin Dublin

Clones Clones

MalinHead Malin Head

Details

The original data set is much larger and was analyzed in detail by Haslett and Raftery (1989). Van Buuren et al (2006) used this subset to investigate the influence of extreme MAR mechanisms on the quality of imputation.

References

Haslett, J. and Raftery, A. E. (1989). *Space-time Modelling with Long-memory Dependence: Assessing Ireland's Wind Power Resource (with Discussion)*. Applied Statistics 38, 1-50. http:http://lib.stat.cmu.edu/datasets/wind.desc http:http://lib.stat.cmu.edu/datasets/wind.data

Van Buuren, S., Brand, J.P.L., Groothuis-Oudshoorn C.G.M., Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, **76**, 12, 1049–1064.

Examples

```
windspeed[1:3,]
```

`with.mids` *Evaluate an expression in multiple imputed datasets*

Description

Performs a computation of each of imputed datasets in data.

Usage

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

Arguments

`data`	An object of type `mids`, which stands for 'multiply imputed data set', typically created by a call to function `mice()`.
`expr`	An expression with a formula object, with the response on the left of a ~ operator, and the terms, separated by + operators, on the right. See the documentation of `lm` and `formula` for details.
`...`	Additional parameters passed to `expr`

Value

A list object of S3 class `mira`

Author(s)

Karin Oudshoorn, Stef van Buuren 2009-2012

References

van Buuren S and Groothuis-Oudshoorn K (2011). `mice`: Multivariate Imputation by Chained Equations in `R`. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mids, mira, pool, pool.compare, pool.r.squared

Examples

```
imp <- mice(nhanes2)
fit1 <- with(data=imp,exp=lm(bmi~age+hyp+chl))
fit2 <- with(data=imp,exp=glm(hyp~age+bmi+chl,family=binomial))
anova.imp <- with(data=imp,exp=anova(lm(bmi~age+hyp+chl)))
```

---

xyplot.mids *Scatterplot of observed and imputed data*

Description

Plotting methods for imputed data using **lattice**. `xyplot()` produces a conditional scatterplots. The function automatically separates the observed (blue) and imputed (red) data. The function extends the usual features of **lattice**.

Usage

```
## S3 method for class 'mids'
xyplot(x, data, na.groups = NULL, groups = NULL,
  as.table = TRUE, theme = mice.theme(), allow.multiple = TRUE,
  outer = TRUE,
  drop.unused.levels = lattice.getOption("drop.unused.levels"), ...,
  subscripts = TRUE, subset = TRUE)
```

Arguments

`x`	A `mids` object, typically created by `mice()` or `mice.mids()`.
`data`	Formula that selects the data to be plotted. This argument follows the **lattice** rules for *formulas*, describing the primary variables (used for the per-panel display) and the optional conditioning variables (which define the subsets plotted in different panels) to be used in the plot. The formula is evaluated on the complete data set in the `long` form. Legal variable names for the formula include `names(x$data)` plus the two administrative factors `.imp` and `.id`. **Extended formula interface:** The primary variable terms (both the LHS `y` and RHS `x`) may consist of multiple terms separated by a '+' sign, e.g., `y1 + y2 ~ x	a * b`. This formula would be taken to mean that the user wants to plot both `y1 ~ x	a * b` and `y2 ~ x	a * b`, but with the `y1 ~ x` and `y2 ~ x` in *separate panels*. This behavior differs from standard **lattice**. *Only combine terms of the same type*, i.e. only factors or only numerical variables. Mixing numerical and categorical data occasionally produces odds labeling of vertical axis.

na.groups	An expression evaluating to a logical vector indicating which two groups are distinguished (e.g. using different colors) in the display. The environment in which this expression is evaluated in the response indicator is.na(x$data). The default na.group = NULL constrasts the observed and missing data in the LHS y variable of the display, i.e. groups created by is.na(y). The expression y creates the groups according to is.na(y). The expression y1 & y2 creates groups by is.na(y1) & is.na(y2), and y1 | y2 creates groups as is.na(y1) | is.na(y2), and so on.
groups	This is the usual groups arguments in **lattice**. It differs from na.groups because it evaluates in the completed data data.frame(complete(x, "long", inc=TRUE)) (as usual), whereas na.groups evaluates in the response indicator. See xyplot for more details. When both na.groups and groups are specified, na.groups takes precedence, and groups is ignored.
theme	A named list containing the graphical parameters. The default function mice.theme produces a short list of default colors, line width, and so on. The extensive list may be obtained from trellis.par.get(). Global graphical parameters like col or cex in high-level calls are still honored, so first experiment with the global parameters. Many setting consists of a pair. For example, mice.theme defines two symbol colors. The first is for the observed data, the second for the imputed data. The theme settings only exist during the call, and do not affect the trellis graphical parameters.
as.table	See xyplot.
outer	See xyplot.
allow.multiple	See xyplot.
drop.unused.levels	See xyplot.
subscripts	See xyplot.
subset	See xyplot.
...	Further arguments, usually not directly processed by the high-level functions documented here, but instead passed on to other functions.

Details

The argument na.groups may be used to specify (combinations of) missingness in any of the variables. The argument groups can be used to specify groups based on the variable values themselves. Only one of both may be active at the same time. When both are specified, na.groups takes precedence over groups.

Use the subset and na.groups together to plots parts of the data. For example, select the first imputed data set by by subset=.imp==1.

Graphical paramaters like col, pch and cex can be specified in the arguments list to alter the plotting symbols. If length(col)==2, the color specification to define the observed and missing groups. col[1] is the color of the 'observed' data, col[2] is the color of the missing or imputed data. A convenient color choice is col=mdc(1:2), a transparent blue color for the observed data, and a transparent red color for the imputed data. A good choice is col=mdc(1:2), pch=20, cex=1.5. These choices can be set for the duration of the session by running mice.theme().

Value

The high-level functions documented here, as well as other high-level Lattice functions, return an object of class "trellis". The update method can be used to subsequently update components of the object, and the print method (usually called by default) will plot it on an appropriate plotting device.

Note

The first two arguments (x and data) are reversed compared to the standard Trellis syntax implemented in **lattice**. This reversal was necessary in order to benefit from automatic method dispatch.

In **mice** the argument x is always a mids object, whereas in **lattice** the argument x is always a formula.

In **mice** the argument data is always a formula object, whereas in **lattice** the argument data is usually a data frame.

All other arguments have identical interpretation.

Author(s)

Stef van Buuren

References

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*, Springer. http://lmdvr.r-forge.r-project.org/

van Buuren S and Groothuis-Oudshoorn K (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, **45**(3), 1-67. http://www.jstatsoft.org/v45/i03/

See Also

mice, stripplot, densityplot, bwplot, Lattice for an overview of the package, as well as xyplot, panel.xyplot, print.trellis, trellis.par.set

Examples

```
require(lattice)

imp <- mice(boys, maxit=1)

### xyplot: scatterplot by imputation number
### observe the erroneous outlying imputed values
### (caused by imputing hgt from bmi)
xyplot(imp, hgt~age|.imp, pch=c(1,20),cex=c(1,1.5))

### same, but label with missingness of wgt (four cases)
xyplot(imp, hgt~age|.imp, na.group=wgt, pch=c(1,20),cex=c(1,1.5))
```

Index