

# Package ‘mice’

January 2, 2012

**Type** Package

**Version** 2.11

**Title** Multivariate Imputation by Chained Equations

**Date** 2011-11-21

**Author** Stef van Buuren <stef.vanbuuren@tno.nl> & Karin Groothuis-Oudshoorn <c.g.m.oudshoorn@utwente.nl>

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

**Depends** R (>= 2.4.0), MASS, nnet, methods, lattice

**Suggests** VIM, lattice, mitools, nlme, Zelig, lme4

**Description** Multiple Imputation using Fully Conditional Specification

**License** GPL-2 | GPL-3

**LazyLoad** yes

**LazyData** yes

**URL** <http://www.stefvanbuuren.nl> ; <http://www.multiple-imputation.com>

**Repository** CRAN

**Date/Publication** 2011-11-22 08:45:27

## R topics documented:

boys . . . . .	2
cbind.mids . . . . .	4
cc . . . . .	6
cci . . . . .	8
ccn . . . . .	9
complete . . . . .	10
glm.mids . . . . .	11

ibind . . . . .	13
lm.mids . . . . .	14
md.pairs . . . . .	15
md.pattern . . . . .	17
mdc . . . . .	18
mice . . . . .	19
mice.impute.2L.norm . . . . .	24
mice.impute.lda . . . . .	26
mice.impute.logreg . . . . .	27
mice.impute.mean . . . . .	28
mice.impute.norm . . . . .	29
mice.impute.norm.boot . . . . .	30
mice.impute.norm.nob . . . . .	31
mice.impute.norm.predict . . . . .	32
mice.impute.passive . . . . .	33
mice.impute.pmm . . . . .	34
mice.impute.polyreg . . . . .	35
mice.impute.sample . . . . .	36
mice.mids . . . . .	37
mids . . . . .	39
mids2mplus . . . . .	40
mids2spss . . . . .	41
mipo . . . . .	43
mira . . . . .	44
nhanes . . . . .	45
nhanes2 . . . . .	46
pool . . . . .	47
pool.compare . . . . .	49
pool.r.squared . . . . .	51
pool.scalar . . . . .	52
popmis . . . . .	54
quickpred . . . . .	55
rbind.mids . . . . .	57
stripplot . . . . .	59
version . . . . .	64
windspeed . . . . .	65
with.mids . . . . .	66
<b>Index</b>	<b>68</b>

---

 boys

*Growth of Dutch boys*


---

### Description

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

**Usage**

```
data(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=2)
z <- complete(imp, 1)

# create imputations for age <8yrs
plot(z$age, z$gen, col=c("blue", "red")[1+is.na(boys$gen)])

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

```

# also, BMI distributions are somewhat different
oldpar <- par(mfrow=c(1,2))
truehist(z$bmi[!is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,
  col="blue",xlab="BMI observed")
truehist(z$bmi[is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,
  col="red",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=2, meth=meth, pred=pred)
z2 <- complete(imp2, 1)

# show that new imputations are consistent
plot(z2$bmi,z2$wgt/(z2$hgt/100)^2, col=c("blue","red")[1+is.na(boys$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="blue",
  xlab="BMI observed")
truehist(z2$bmi[is.na(boys$bmi)],h=1,xlim=c(10,30),ymax=0.25,col="red",
  xlab="BMI imputed")
par(oldpar)

```

---

cbind.mids

*Combine a Multiply Imputed Data Set with other mids object or dataframe*


---

## Description

Columnwise combination of mids objects

## Usage

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

## Arguments

x	A mids object.
y	A mids object or a dataframe, matrix, factor or vector.
...	Dataframes, matrices, vectors or factors. These can be given as named arguments.

## Details

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 dataframe 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.

## Value

<code>call</code>	A vector, with first argument the <code>mice()</code> statement that created <code>x</code> and second argument the call to <code>cbind.mids()</code> .
<code>data</code>	The <code>cbind</code> of the (incomplete) data in <code>x\$data</code> and <code>y\$data</code> .
<code>m</code>	The number of imputations.
<code>nmis</code>	An array containing the number of missing observations per column.
<code>imp</code>	A list of <code>nvar</code> components with the generated multiple imputations. Each part of the list is a <code>nmis[j]</code> by <code>m</code> matrix of imputed values for variable <code>j</code> . The original data of <code>y</code> will be copied into this list, including the missing values of <code>y</code> then <code>y</code> is not imputed.
<code>method</code>	A vector of strings of length( <code>nvar</code> ) specifying the elementary imputation method per column. If <code>y</code> is a <code>mids</code> object this vector is a combination of <code>x\$method</code> and <code>y\$method</code> , otherwise this vector is <code>x\$method</code> and for the columns of <code>y</code> the method is set to <code>""</code> .
<code>predictorMatrix</code>	A square matrix of size <code>ncol(data)</code> containing code 0/1 data specifying the predictor set. If <code>x</code> and <code>y</code> are <code>mids</code> objects then the predictor matrices of <code>x</code> and <code>y</code> are combined with zero matrices on the off diagonal blocks. Otherwise the variables in <code>y</code> are included in the predictor matrix of <code>x</code> such that <code>y</code> is not used as predictor(s) and not imputed as well.
<code>visitSequence</code>	The sequence in which columns are visited. The same as <code>x\$visitSequence</code> .
<code>seed</code>	The seed value of the solution, <code>x\$seed</code> .
<code>iteration</code>	Last Gibbs sampling iteration number, <code>x\$iteration</code> .
<code>lastSeedValue</code>	The most recent seed value, <code>x\$lastSeedValue</code>
<code>chainMean</code>	Combination of <code>x\$chainMean</code> and <code>y\$chainMean</code> . If <code>y\$chainMean</code> does not exist this element equals <code>x\$chainMean</code> .
<code>chainVar</code>	Combination of <code>x\$chainVar</code> and <code>y\$chainVar</code> . If <code>y\$chainVar</code> does not exist this element equals <code>x\$chainVar</code> .
<code>pad</code>	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 <code>x\$pad</code> and <code>y\$pad</code> if <code>y</code> is a <code>mids</code> object. Otherwise, it is defined by the settings of <code>x</code> and the combination of the data <code>x\$data</code> and <code>y</code> .

Remark that 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)
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=2)
impb <- mice(temp2, maxit=3)

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

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

**Description**

Extracts complete and incomplete cases

**Usage**

```
## S4 method for signature 'data.frame'
cc(x, drop = TRUE)
## S4 method for signature 'matrix'
cc(x, drop = TRUE)
## S4 method for signature 'mids'
cc(x, drop = TRUE)
## S4 method for signature 'data.frame'
ic(x, drop = TRUE)
## S4 method for signature 'matrix'
ic(x, drop = TRUE)
## S4 method for signature 'mids'
ic(x, drop = TRUE)
```

**Arguments**

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

**Details**

This is the same as listwise deletion. `cc()` is equivalent to `na.omit()`. Missing values in `x` are coded as `NA`.

**Value**

A vector, matrix of `data.frame` containing the data of the complete cases (`cc`) or the incomplete cases (`ic`).

**Author(s)**

Stef van Buuren, 2010.

**See Also**

[na.omit](#), [cci](#), [ici](#), [codeccn](#), [codelinkicn](#)

**Examples**

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

---

`cci`*Extracts (in)complete case indicator*

---

## Description

Extracts (in)complete case indicator

## Usage

```
## S4 method for signature 'data.frame'  
cci(x)  
## S4 method for signature 'matrix'  
cci(x)  
## S4 method for signature 'mids'  
cci(x)  
## S4 method for signature 'data.frame'  
ici(x)  
## S4 method for signature 'matrix'  
ici(x)  
## S4 method for signature 'mids'  
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.

## Details

This array is useful for extracting subsets of the complete and incomplete data. Missing values in `x` are coded as `NA`.

## Value

A logical vector indicating the complete and the incomplete cases, with a length of `nrow(x)` if `x` is a `data.frame` or `matrix`, and with length `length(x)` in other cases.

## Author(s)

Stef van Buuren, 2010.

## See Also

[na.omit](#), [cc](#), [ic](#), [codeccn](#), [codelinkicn](#)

**Examples**

```

cci(nhanes) # indicator for 13 complete cases
ici(nhanes) # indicator for 12 rows with incomplete 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

*Number of (in)complete cases***Description**

Calculates the number of (in)complete cases

**Usage**

```

## S4 method for signature 'data.frame'
ccn(x)
## S4 method for signature 'matrix'
ccn(x)
## S4 method for signature 'mids'
ccn(x)
## S4 method for signature 'data.frame'
icn(x)
## S4 method for signature 'matrix'
icn(x)
## S4 method for signature 'mids'
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 (in)complete data.

**Author(s)**

Stef van Buuren, 2010.

**See Also**

[cc](#), [ic](#), [codecci](#), [codelinkici](#)

**Examples**

```
ccn(nhanes) # 13 complete cases
icn(nhanes) # the remaining 12 rows
icn(nhanes[,c("bmi","hyp")]) # number of cases with incomplete bmi and hyp
```

complete

*Creates a Complete Flat File from a Multiply Imputed Data Set***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**

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

**Details**

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

"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.

"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.

"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 imputed 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)
```

---

glm.mids

*Generalized Linear Model for Multiply Imputed Data*

---

**Description**

Applies glm() to a multiply imputed data set

**Usage**

```
## S3 method for class 'mids'
glm(formula, family = gaussian, data, ...)
```

**Arguments**

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

**Details**

This function is included for backward compatibility with V1.0. The function is superseded 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. <http://www.stefvanbuuren.nl/publications/MICEV1.0ManualTNO000382000.pdf>

**See Also**

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

**Examples**

```
imp <- mice(nhanes)
glm.mids((hyp==2)~bmi+chl, data=imp)
# fit
# $call:
# glm.mids(formula = (hyp == 2) ~ bmi + chl, data = imp)
#
# $call1:
# mice(data = nhanes)
#
# $nmis:
# age bmi hyp chl
# 0 9 8 10
#
# $analyses:
# $analyses[[1]]:
```

```
# Call:
# glm(formula = formula, data = data.i)
#
# Coefficients:
# (Intercept)      bmi      chl
# -0.4746337 -0.01565534 0.005417846
#
# Degrees of Freedom: 25 Total; 22 Residual
# Residual Deviance: 2.323886
#
# $analyses[[2]]:
# Call:
# glm(formula = formula, data = data.i)
#
# Coefficients:
# (Intercept)      bmi      chl
# -0.1184695 -0.02885779 0.006090282
#
# Degrees of Freedom: 25 Total; 22 Residual
# Residual Deviance: 3.647927
#
```

---

ibind

*Combine imputations fitted to the same data*

---

## Description

Combine imputations fitted to the same data

## Usage

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

## Arguments

x	A mids object.
y	A mids object.

## Details

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.

**Value**

call	A vector, with first argument the mice statement that created x and second argument the call to ibind().
data	The incomplete data in x and y.
m	Defined as $x\$m+y\$m$ , the total number of imputations from x and y.
nmis	Defined as $x\$nmis$ , an array containing the number of missing observations per column of $x\$data$ .
imp	A combination of $x\$imp$ and $y\$imp$ .
method	Defined as $x\$method$ .
predictorMatrix	Defined as $x\$predictorMatrix$ .
visitSequence	$x\$visitSequence$
seed	Defined as $x\$seed$ .
iteration	Last Gibbs sampling iteration number, $x\$iteration$ .
lastSeedValue	Defined as $x\$lastSeedValue$ .
chainMean	Combination of $x\$chainMean$ and $y\$chainMean$ .
chainVar	Combination of $x\$chainVar$ and $y\$chainVar$ .
pad	Defined as $x\$pad$ (which should equal $y\$pad$ ).

**Author(s)**

Karin Groothuis-Oudshoorn, Stef van Buuren, 2009

**See Also**

[rbind.mids](#), [cbind.mids](#), [mids](#)

---

lm.mids

---

*Linear Regression on Multiply Imputed Data*


---

**Description**

Applies `lm()` to multiply imputed data set

**Usage**

```
## S3 method for class 'mids'
lm(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 <code>lm</code> and <code>formula</code> for details.
data	An object of type 'mids', which stands for 'multiply imputed data set', typically created by a call to function <code>mice()</code> .
...	Additional parameters passed to <code>lm</code>

**Details**

This function is included for backward compatibility with V1.0. The function is superseded 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)
```

---

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:

rr response-response, both variables are observed

rm response-missing, row observed, column missing

mr missing -response, row missing, column observed

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	<i>Missing Data Pattern</i>
------------	-----------------------------

---

## 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  $\text{ncol}(x)+1$  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")
```

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

### Arguments

<code>r</code>	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, <code>r</code> may contain the strings "observed", "missing", or "both", or abbreviations thereof.
<code>s</code>	A character vector containing the strings "symbol" or "line", or abbreviations thereof.
<code>transparent</code>	A logical indicating whether alpha-transparency is allowed. The default is TRUE.
<code>alpha.fill</code>	A numerical values between 0 and 1 that indicates the default alpha value for fills.
<code>cso</code>	The symbol color for the observed data. The default is a transparent blue.
<code>csi</code>	The symbol color for the missing or imputed data. The default is a transparent red.
<code>csc</code>	The symbol color for the combined observed and imputed data. The default is a grey color.
<code>clo</code>	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. `mice.theme` return 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, 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)],
      post = vector("character", length = ncol(data)),
      defaultMethod = c("pmm", "logreg", "polyreg", "polr"),
      maxit = 5,
      diagnostics = TRUE,
      printFlag = TRUE,
      seed = NA,
      imputationMethod = NULL,
      defaultImputationMethod = NULL,
      ...
)

```

**Arguments**

<code>data</code>	A data frame or a matrix containing the incomplete data. Missing values are coded as NA.
<code>m</code>	Number of multiple imputations. The default is <code>m=5</code> .
<code>method</code>	Can be either a single string, or a vector of strings with length <code>ncol(data)</code> , 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 <code>defaultMethod</code> argument. Columns that need not be imputed have the empty method <code>""</code> . See details for more information.
<code>predictorMatrix</code>	A square matrix of size <code>ncol(data)</code> 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 <code>predictorMatrix</code> must be zero. The default for <code>predictorMatrix</code> 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.
<code>visitSequence</code>	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 <code>1:ncol(data)</code> 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.

<code>post</code>	A vector of strings with length <code>ncol(data)</code> , specifying expressions. Each string is parsed and executed within the <code>sampler()</code> function to postprocess imputed values. The default is to do nothing, indicated by a vector of empty strings <code>""</code> .
<code>defaultMethod</code>	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: <code>pmm</code> , predictive mean matching (numeric data) <code>logreg</code> , logistic regression imputation (binary data, factor with 2 levels) <code>polyreg</code> , polytomous regression imputation for unordered categorical data (factor $\geq 2$ levels) <code>polr</code> , proportional odds model for (ordered, $\geq 2$ levels)
<code>maxit</code>	A scalar giving the number of iterations. The default is 5.
<code>diagnostics</code>	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.
<code>printFlag</code>	If TRUE, mice will print history on console. Use <code>print=FALSE</code> for silent computation.
<code>seed</code>	An integer that is used as argument by the <code>set.seed()</code> for offsetting the random number generator. Default is to leave the random number generator alone.
<code>imputationMethod</code>	Same as <code>method</code> argument. Included for backwards compatibility.
<code>defaultImputationMethod</code>	Same as <code>defaultMethod</code> argument. Included for backwards compatibility.
<code>...</code>	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 (numeric)

`norm` Bayesian linear regression (numeric)

`norm.nob` Linear regression ignoring model error (numeric)

mean Unconditional mean imputation (numeric)  
 2l.norm Two-level normal imputation (numeric)  
 logreg Logistic regression (factor, 2 levels)  
 polyreg Polytomous logistic regression (factor, >= 2 levels)  
 polr Proportional odds model (ordered, >=2 levels)  
 lda Linear discriminant analysis (factor, >= 2 categories)  
 sample Random sample from the observed values (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 occurrence of `paste("mice.impute.", method[j], sep="")` in the search path. The mechanism allows users 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 deterministic 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 object of class `mids` (multiply imputed data set) with components

<code>call</code>	The call that created the object
<code>data</code>	A copy of the incomplete data set
<code>m</code>	The number of imputations
<code>nmis</code>	An array of length <code>ncol(data)</code> containing the number of missing observations per column

<code>imp</code>	A list of <code>ncol(data)</code> components with the generated multiple imputations. Each part of the list is a <code>nmis[j]</code> by <code>m</code> matrix of imputed values for variable <code>data[,j]</code> . The component equals <code>NULL</code> for columns without missing data.
<code>method</code>	A vector of strings of length <code>ncol(data)</code> specifying the elementary imputation method per column
<code>predictorMatrix</code>	A square matrix of size <code>ncol(data)</code> containing 0/1 data specifying the predictor set
<code>visitSequence</code>	The sequence in which columns are visited
<code>post</code>	A vector of strings of length <code>ncol(data)</code> with commands for post-processing
<code>seed</code>	The seed value of the solution
<code>iteration</code>	Last Gibbs sampling iteration number
<code>lastSeedValue</code>	The most recent seed value
<code>chainMean</code>	An array containing the mean of the generated multiple imputations. The array can be used for monitoring convergence. Factors are replaced by their numerical representation using <code>as.integer()</code> . Note that observed data are not present in this mean.
<code>chainVar</code>	An array with similar structure of <code>chainMean</code> , containing the variances of the imputed values.
<code>pad</code>	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 <code>pad\$data</code> (data padded with columns for factors), <code>pad\$predictorMatrix</code> (predictor matrix for the padded data), <code>pad\$method</code> (imputation methods applied to the padded data), the vector <code>pad\$visitSequence</code> (the visit sequence applied to the padded data), <code>pad\$post</code> (post-processing commands for padded data) and <code>categories</code> (a matrix containing descriptive information about the padding operation).
<code>loggedEvents</code>	A matrix with six columns containing a record of automatic removal actions. It is <code>NULL</code> if 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 <code>polyreg</code> . Column <code>it</code> is the iteration number at which the record was added, <code>im</code> is the imputation number, <code>co</code> is the column number in the data, <code>dep</code> is the name of the name of the dependent variable, <code>meth</code> is the imputation method used, and <code>out</code> is a (possibly long) character vector with the names of the altered or removed predictors.

### Author(s)

Stef van Buuren ([stef.vanbuuren@tno.nl](mailto:stef.vanbuuren@tno.nl)), Karin Groothuis-Oudshoorn ([c.g.m.oudshoorn@utwente.nl](mailto:c.g.m.oudshoorn@utwente.nl)) (2000-2010) with contributions of Roel de Jong, Jason Turner, 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., 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. <http://www.stefvanbuuren.nl/publications/FCSinmultivariateimputation-JSCS2006.pdf>
- Van Buuren, S. (2007) Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, **16**, 3, 219–242. <http://www.stefvanbuuren.nl/publications/MIbyFCS-SMMR2007.pdf>
- 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. <http://www.stefvanbuuren.nl/publications/Multipleimputation-StatMed1999.pdf>
- 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

[complete](#), [mids](#), [with.mids](#), [set.seed](#)

## 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, ...)
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()`.

**Value**

A vector of length `nmi`s with imputations.

**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 Roberts, J.K. (Eds.), *The Handbook of Advanced Multilevel Analysis*, Chapter 10, pp. 173–196. Milton Park, UK: Routledge.

---

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 `nmi.s` 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      *Multiple Imputation by Logistic Regression*

---

## Description

Imputes univariate missing data using logistic regression.

## Usage

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

## Details

Imputation for binary response variables by the Bayesian logistic regression model (Rubin 1987, p. 169-170) or bootstrap logistic regression model. 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.  $\text{logit}^{-1}(X \text{ BETA})$
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. The bootstrap method draws a bootstrap sample from `y[ry]` and `x[ry,]`. Perfect prediction is handled by the data augmentation method.

**Value**

imp                    A vector of length `nmi`s 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 `nmi`s 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 nmi s 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 bootstrap

**Usage**

```
mice.impute.norm.boot(y, ry, x, ridge=0.00001, ...)
```

**Arguments**

<code>y</code>	Incomplete data vector of length <code>n</code>
<code>ry</code>	Vector of missing data pattern (FALSE=missing, TRUE=observed)
<code>x</code>	Matrix ( <code>n x p</code> ) of complete covariates.
<code>ridge</code>	Ridge parameter
<code>...</code>	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 `n` `m` `s` 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 nmi s 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=0.00001, ...)
```

**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 return predicted values to as imputations. The ridge parameter adds a penalty term  $\text{ridge} \cdot \text{diag}(x^T x)$  to the variance-covariance matrix  $x^T x$ .

**Value**

A vector of length `n` 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**

t	The transformed data
---	----------------------

**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, ...)
```

### Arguments

y	Numeric vector with incomplete data
ry	Response pattern of y (TRUE=observed, FALSE=missing)
x	Design matrix with $\text{length}(y)$ rows and p columns containing complete covariates.
...	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  $y_{\text{obs}}\beta^*$  and  $y_{\text{mis}}\beta^*$
4. For each  $y_{\text{mis}}$ , find the observation with closest predicted value, and take its observed value in y as the imputation.
5. If there is more than one candidate, make a random draw among them. Note: The matching is done on predicted y, NOT on observed y.

### Value

imp      Numeric vector of length  $\text{sum}(!ry)$  with imputations

### Author(s)

Stef van Buuren, Karin Groothuis-Oudshoorn, 2000

## 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. <http://www.stefvanbuuren.nl/publications/FCSinmultivariateimputation-JSCS2006.pdf>
- 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.polyreg     *Imputation by Polytomous Regression*

---

## 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, ...)
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, factors with more than two levels are imputed by `mice.impute.polyreg` (for unordered factors) and `mice.impute.polr` (for ordered factors).

The function `mice.impute.polyreg` imputation for 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.

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, both algorithms 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 `nmi.s` with imputations.

### Author(s)

Stef van Buuren, Karin Groothuis-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.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 `nmi` s 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**

```
## S3 method for class 'mids'
mice(obj, maxit=1, diagnostics=TRUE, printFlag=TRUE, ...)
```

**Arguments**

obj	An object of class <code>mids</code> , typically produced by a previous call to <code>mice()</code> or <code>mice.mids()</code>
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](#)

## 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
#
```

---

mids

*Multiply Imputed Data Set*


---

### Description

An object containing 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`.

### Usage

```
is.mids(x)
## S4 method for signature 'mids'
print(x,...)
## S4 method for signature 'mids'
summary(object,...)
## S4 method for signature 'mids,ANY'
plot(x, y, ...)

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

### Arguments

<code>x</code> , <code>object</code>	A object of class <code>mids</code> .
<code>y</code>	A character vector containing variable names, an integer vector of indices of imputed variables, a logical vector of length( <code>dimnames(x\$chainMean[, , 1])[[1]]</code> ), or a formula. The result of the evaluation will be plotted in the trace plot.
<code>theme</code>	List of settings with selected graphical parameters to control the <b>lattice</b> function <code>xyplot()</code> .
<code>layout</code>	Vector of two numbers controlling the number of panels in horizontal and vertical direction, respectively.
<code>type</code>	Plot type parameter.
<code>col</code>	Color parameter.
<code>lty</code>	Line type parameter.
<code>...</code>	Currently not used.

### Value

<code>call</code>	The call that created the object.
<code>data</code>	A copy of the incomplete data set.
<code>m</code>	The number of imputations.
<code>nmis</code>	An array containing the number of missing observations per column.

imp	A list of nvar components with the generated multiple imputations. Each part of the list is a <code>nmis[j]</code> by <code>m</code> matrix of imputed values for variable <code>j</code> .
method	A vector of strings of length( <code>nvar</code> ) specifying the elementary imputation method per column.
predictorMatrix	A square matrix of size <code>ncol(data)</code> containing code 0/1 data specifying the predictor set.
visitSequence	The sequence in which columns are visited.
post	A vector of strings of length <code>ncol(data)</code> 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 <code>m</code> components. Each component is a <code>length(visitSequence)</code> by <code>maxit</code> 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 <code>chainMean</code> , containing the covariances of the imputed values.
pad	A list containing various settings of the padded imputation model, i.e. the imputation model after creating dummy variables. Normally, this array is only useful for error checking.

**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 Multiply Imputed Data 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**

<code>imp</code>	The <code>imp</code> argument is an object of class <code>mids</code> , typically produced by the <code>mice()</code> function.
<code>file.prefix</code>	A character string describing the prefix of the output data files.
<code>path</code>	A character string containing the path of the output file. By default, files are written to the current R working directory.
<code>sep</code>	The separator between the data fields.
<code>dec</code>	The decimal separator for numerical data.
<code>silent</code>	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 Multiply Imputed Data 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**

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

*Multiply Imputed Pooled Analysis*


---

### Description

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

### Usage

```
is.mipo(x)
## S4 method for signature 'mipo'
print(x,...)
## S4 method for signature 'mipo'
summary(object,...)
```

### Arguments

<code>x</code> , <code>object</code>	An object of class <code>mira</code> containing the <code>m</code> fit objects of a complete data analysis, plus some additional information.
<code>...</code>	not used.

### Value

<code>call</code>	The call that created the mipo object.
<code>call1</code>	The call that created the mira object that was used in call.
<code>call2</code>	The call that created the mids object that was used in call1.
<code>nmis</code>	An array containing the number of missing observations per column.
<code>m</code>	Number of multiple imputations.
<code>qhat</code>	An <code>m</code> by <code>npar</code> matrix containing the complete data estimates for the <code>npar</code> parameters of the <code>m</code> complete data analyses.
<code>u</code>	An <code>m</code> by <code>npar</code> by <code>npar</code> array containing the variance-covariance matrices of the <code>m</code> complete data analyses.
<code>qbar</code>	The average of complete data estimates.
<code>ubar</code>	The average of the variance-covariance matrix of the complete data estimates.
<code>b</code>	The between imputation variance-covariance matrix.
<code>t</code>	The total variance-covariance matrix.
<code>r</code>	Relative increases in variance due to missing data.
<code>dfcom</code>	Degrees of freedom in the hypothetically complete data: the sample size minus the number of free parameters.
<code>df</code>	Degrees of freedom associated with the t-statistics.
<code>fmi</code>	Fraction of missing information.
<code>lambda</code>	Proportion of the variation attributable to the missing data: $(b+b/m)/t$ .

**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

*Multiply Imputed Repeated Analyses*

---

**Description**

The mira object is generated by the `with.mids()`, `lm.mids()` and `glm.mids()` functions. 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`.

**Usage**

```
is.mira(x)
as.mira(fitlist)
## S4 method for signature 'mira'
print(x)
## S4 method for signature 'mira'
summary(object)
```

**Arguments**

<code>x</code> , <code>object</code>	An object containing the <code>m</code> fit objects of a complete data analysis, plus some additional information.
<code>fitlist</code>	An list of fitted objects, where each list element is a fit object. This can, for example, be produced by the <code>by()</code> function.

**Value**

<code>call</code>	The call that created the object.
<code>call1</code>	The call that created the mids object that was used in <code>call</code> .
<code>nmis</code>	An array containing the number of missing observations per column.
<code>analyses</code>	A list of <code>m</code> components containing the individual fit objects from each of the <code>m</code> complete data analyses.

**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](#)

---

nhanes

*NHANES example - all variables numerical*

---

**Description**

A small data set with non-monotone missing values.

**Usage**

```
data(nhanes)
```

**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.

## Usage

```
data(nhanes2)
```

## 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
```

---

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 <code>mira</code> , produced by <code>with.mids()</code> .
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'. The object is a list containing the following items:

call	The call to the pool function.
call1	The original call how the <code>mira</code> object was calculated.
call2	The original call to the <code>mice</code> function to calculate the underlying <code>midsobject</code> .
formula	The formula that is used in <code>call1</code> .
nmis	The number of missing entries for each variable.
m	The number of imputations
qhat	A matrix, containing the estimated coefficients of the $m$ repeated complete data analyses

u	The corresponding $m$ variance matrices of the estimates in an three dimensional array.
qbar	The pooled estimate, formula (3.1.2) Rubin (1987).
ubar	The mean of the variances, formula (3.1.3), Rubin (1987).
b	The within imputation variance, formula (3.1.4), Rubin (1987).
t	Total variance of the pooled estimates, formula (3.1.5), Rubin (1987).
r	Relative increase in variance due to nonresponse, formula (3.1.7), Rubin (1987).
dfcom	Degrees of freedom for estimates in the complete data analysis.
df	Degrees of freedom for t reference distribution, calculated according to the article of Barnard and Rubin (1999).
fmi	Fraction missing information due to nonresponse, formula (3.1.10), Rubin (1987).
gamma	Proportion of the total variance explained by the imputations.

**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](#), [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

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

call	The call to the pool.compare function
call11	The call that created fit1
call12	The call that created the imputations.
call01	The call that created fit0
call02	The call that created the imputations.
method	The method used to compare two models: "Wald" or "likelihood"
nmis	The number of missing entries for each variable.
m	The number of imputations
qhat1	A matrix, containing the estimated coefficients of the $m$ repeated complete data analyses from fit1
qhat0	A matrix, containing the estimated coefficients of the $m$ repeated complete data analyses from fit0
ubar1	The mean of the variances of object1, formula (3.1.3), Rubin (1987).
ubar0	The mean of the variances of object0, formula (3.1.3), Rubin (1987).
qbar1	The pooled estimate of object1, formula (3.1.2) Rubin (1987).
qbar0	The pooled estimate of object0, formula (3.1.2) Rubin (1987).
Dm	The test statistic
rm	Relative increase in variance due to nonresponse, formula (3.1.7), Rubin (1987).
df1	df1; Under the null hypothesis it is assumed that Dm has an F distribution with (df1,df2) degrees of freedom.
df2	df2
pvalue	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).
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)

```

---

pool.r.squared	<i>Pooling: R squared</i>
----------------	---------------------------

---

**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 <code>lm.mids</code> or <code>with.mids</code> with <code>lm</code> as modelling function.
adjusted	A logical value. If <code>adjusted=TRUE</code> then the adjusted $R^2$ is calculated. The default value is <code>FALSE</code> .

**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 elements:

est	The pooled $R^2$ estimate
lo95	The 95 % lower bound of the pooled $R^2$ .

hi95            The 95 % upper bound of the pooled  $R^2$ .  
 fmi            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* (to appear).

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)
```

**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.

**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

m                    The number of imputations  
qhat                The m univariate estimates of repeated complete data analyses  
u                    The corresponding m variances of the univariate estimates  
qbar                The pooled univariate estimate, formula (3.1.2) Rubin (1987).  
ubar                The mean of the variances, formula (3.1.3) Rubin (1987).  
b                    The within imputation variance, formula (3.1.4) Rubin (1987).  
t                    Total variance of the pooled estimated, formula (3.1.5) Rubin (1987).  
r                    Relative increase in variance due to nonresponse, formula (3.1.7) Rubin (1987).  
df                  Degrees of freedom for t reference distribution, formula (3.1.6) Rubin (1987).  
f                    Fraction missing information due to nonresponse, formula (3.1.10) Rubin (1987).

**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)
}
pool.scalar(Q,U)

#pool.scalar(Q,U)
#$m
#[1] 5
#
#$qhat
#[1] 26.764 26.748 27.024 27.340 26.436
#
#$u
#[1] 17.85490 19.11677 20.61440 21.05750 15.16990
#
#$qbar
#[1] 26.8624
#
#$ubar
#[1] 18.76269
#
#$b
#[1] 0.1147008
#
#t
#[1] 18.90033
#
#$r
#[1] 0.007335885
#
#$df
#[1] 75422.96
#
#$f
#[1] 0.007308785
#

```

---

popmis

*Hox pupil popularity data with missing popularity scores*


---

**Description**

Hox pupil popularity data with some missing popularity scores

**Usage**

```
data(popmis)
```

**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,]
```

---

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**

<code>data</code>	Matrix or data frame with incomplete data.
<code>mincor</code>	A scalar, numeric vector (of size <code>ncol(data)</code> ) or numeric matrix (square, of size <code>ncol(data)</code> ) specifying the minimum threshold(s) against which the absolute correlation in the data is compared.
<code>minpuc</code>	A scalar, vector (of size <code>ncol(data)</code> ) or matrix (square, of size <code>ncol(data)</code> ) specifying the minimum threshold(s) for the proportion of usable cases.
<code>include</code>	A string or a vector of strings containing one or more variable names from <code>names(data)</code> . Variables specified are always included as a predictor.
<code>exclude</code>	A string or a vector of strings containing one or more variable names from <code>names(data)</code> . Variables specified are always excluded as a predictor.
<code>method</code>	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 auxiliary, 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. <http://www.stefvanbuuren.nl/publications/Multipleimputation-StatMed1999.pdf>

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	<i>Combine a Multiply Imputed Data Set with other mids object or dataframe</i>
------------	--

---

## Description

Append mids objects by rows

## Usage

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

**Arguments**

<code>x</code>	A <code>mids</code> object.
<code>y</code>	A <code>mids</code> object or a dataframe, matrix, factor or vector.
<code>...</code>	Dataframes, matrices, vectors or factors. 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**

<code>call</code>	A vector, with first argument the <code>mice()</code> statement that created <code>x</code> and second argument the call to <code>rbind.mids()</code>
<code>data</code>	The rowwise combination of the (incomplete) data in <code>x</code> and <code>y</code> .
<code>m</code>	<code>x\$m</code>
<code>nmis</code>	An array containing the number of missing observations per column, defined as <code>x\$nmis + y\$nmis</code>
<code>imp</code>	A list of <code>nvar</code> components with the generated multiple imputations. Each part of the list is a <code>nmis[j]</code> by <code>m</code> matrix of imputed values for variable <code>j</code> . If <code>y</code> is a <code>mids</code> object then <code>imp[[j]]</code> equals <code>rbind(x\$imp[[j]], y\$imp[[j]])</code> ; otherwise the original data of <code>y</code> will be copied into this list, including the missing values of <code>y</code> then <code>y</code> is not imputed.
<code>method</code>	A vector of strings of length( <code>nvar</code> ) specifying the elementary imputation method per column defined as <code>x\$method</code>
<code>predictorMatrix</code>	A square matrix of size <code>ncol(data)</code> containing code 0/1 data specifying the predictor set defined as <code>x\$predictorMatrix</code>
<code>visitSequence</code>	The sequence in which columns are visited, defined as <code>x\$visitSequence</code> .
<code>seed</code>	The seed value of the solution, <code>x\$seed</code>
<code>iteration</code>	Last Gibbs sampling iteration number, <code>x\$iteration</code>
<code>lastSeedValue</code>	The most recent seed value, <code>x\$lastSeedValue</code>
<code>chainMean</code>	Set to NA
<code>chainVar</code>	Set to NA
<code>pad</code>	<code>x\$pad</code> , a list containing various settings of the padded imputation model, i.e. the imputation model after creating dummy variables

**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](#)

---

stripplot	<i>Box-and-whisker plot, stripplot, density plot and scatterplot for imputed data</i>
-----------	---

---

## Description

Plotting methods for imputed data using **lattice**. `bwplot` produces box-and-whisker plots, `stripplot` produces one-dimensional scatterplots, `densityplot` produces plots of the densities, and `xyplot` produces a conditional scatterplots. Each function automatically separates the observed and imputed data in a natural way. 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)

## 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)

## 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)

## 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 <code>mids</code> object, typically created by <code>mice()</code> or <code>mice.mids()</code> .
data	<p>Formula that selects the data to be plotted. This argument follows the <b>lattice</b> rules for <i>formulas</i>, 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.</p> <p>The formula is evaluated on the complete data set in the long form. Legal variable names for the formula include <code>names(x\$data)</code> plus the two administrative factors <code>.imp</code> and <code>.id</code>.</p> <p><b>Extended formula interface:</b> The primary variable terms (both the LHS <code>y</code> and RHS <code>x</code>) may consist of multiple terms separated by a '+' sign, e.g., <code>y1 + y2 ~ x   a * b</code>. This formula would be taken to mean that the user wants to plot both <code>y1 ~ x   a * b</code> and <code>y2 ~ x   a * b</code>, but with the <code>y1 ~ x</code> and <code>y2 ~ x</code> in <i>separate panels</i>. This behavior differs from standard <b>lattice</b>. <i>Only combine terms of the same type</i>, i.e. only factors or only numerical variables. Mixing numerical and categorical data occasionally produces odds labeling of vertical axis.</p> <p>For convenience, in <code>stripplot()</code> and <code>bwplot</code> the formula <code>y~.imp</code> may be abbreviated as <code>y</code>. This applies only to a single <code>y</code>, and does not (yet) work for <code>y1+y2~.imp</code>.</p> <p>The function <code>densityplot</code> does not use the <code>y</code> terms in the formula. Density plots for <code>x1</code> and <code>x2</code> are requested as <code>~ x1 + x2</code>.</p>
na.groups	<p>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 <code>is.na(x\$data)</code>.</p> <p>The default <code>na.group = NULL</code> contrasts the observed and missing data in the LHS <code>y</code> variable of the display, i.e. groups created by <code>is.na(y)</code>. The expression <code>y</code> creates the groups according to <code>is.na(y)</code>. The expression <code>y1 &amp; y2</code> creates groups by <code>is.na(y1) &amp; is.na(y2)</code>, and <code>y1   y2</code> creates groups as <code>is.na(y1)   is.na(y2)</code>, and so on.</p>
groups	<p>This is the usual <code>groups</code> arguments in <b>lattice</b>. It differs from <code>na.groups</code> because it evaluates in the completed data <code>data.frame(complete(x, "long", inc=TRUE))</code> (as usual), whereas <code>na.groups</code> evaluates in the response indicator. See <a href="#">xyplot</a> for more details. When both <code>na.groups</code> and <code>groups</code> are specified, <code>na.groups</code> takes precedence, and <code>groups</code> is ignored.</p>
plot.points	A logical used in <code>densityplot</code> that signals whether the points should be plotted.
theme	<p>A named list containing the graphical parameters. The default function <code>mice.theme</code> produces a short list of default colors, line width, and so on. The extensive list may be obtained from <code>trellis.par.get()</code>. Global graphical parameters like <code>col</code> or <code>cex</code> in high-level calls are still honored, so first experiment with the global parameters. Many setting consists of a pair. For example, <code>mice.theme</code> 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.</p>
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 <code>densityplot</code> . Multiplication factor of the line width of the observed density. <code>thicker=1</code> uses the same thickness for the observed and imputed data.
jitter.data	See <a href="#">panel.xyplot</a> .
horizontal	See <a href="#">xyplot</a> .
as.table	See <a href="#">xyplot</a> .
panel	See <a href="#">xyplot</a> .
default.prepanel	See <a href="#">xyplot</a> .
outer	See <a href="#">xyplot</a> .
allow.multiple	See <a href="#">xyplot</a> .
drop.unused.levels	See <a href="#">xyplot</a> .
subscripts	See <a href="#">xyplot</a> .
subset	See <a href="#">xyplot</a> .
...	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 `subset=.imp==1`.

Graphical parameters 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](#), [Lattice](#) for an overview of the package, as well as [xyplot](#), [densityplot](#), [panel.bwplot](#), [panel.stripplot](#), [panel.densityplot](#), [panel.xyplot](#), [print.trellis](#), [trellis.par.set](#)

**Examples**

```
imp <- mice(boys, maxit=2)

### 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())

### stripplot, all numerical variables
stripplot(imp)

### same, but with improved display
stripplot(imp, col=c("grey",mdc(2)),pch=c(1,20))
```

```

### distribution per imputation of height, weight and bmi
### labeled by their own missingness
stripplot(imp, hgt+wgt+bmi~.imp, cex=c(2,4), pch=c(1,20),jitter=FALSE,
layout=c(3,1))

### same, but labeled with the missingness of wgt (just four cases)
stripplot(imp, hgt+wgt+bmi~.imp, na=wgt, cex=c(2,4), pch=c(1,20),jitter=FALSE,
layout=c(3,1))

### 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
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
stripplot(imp, gen~phb, factor=2, cex=c(8,1), hor=TRUE)

### circle fun
stripplot(imp, gen~.imp, factor=2, cex=c(8,6), hor=FALSE, na=wgt,outer=TRUE,scales="free",pch=c(1,19))

### 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)

### The more powerful density plot of all
### numerical variables with at least
### two missing values.
densityplot(imp)

### 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))

```

**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")
```

---

windspeed	<i>Subset of Irish wind speed data</i>
-----------	--

---

**Description**

Subset of Irish wind speed data

**Usage**

```
data(windspeed)
```

**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://lib.stat.cmu.edu/datasets/wind.desc> <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. <http://www.stefvanbuuren.nl/publications/FCSinmultivariateimputation-JSCS2006.pdf>

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

**Value**

call	The call that created the object.
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.
formula	The formula of the call that created the object.

**Author(s)**

Karin Oudshoorn, 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**

[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)))
```

# Index

- \*Topic **MANIP**
  - mids2mplus, 40
  - mids2spss, 41
- \*Topic **classes**
  - mids, 39
  - mipo, 43
  - mira, 44
- \*Topic **datagen**
  - mice.impute.2L.norm, 24
  - mice.impute.lda, 26
  - mice.impute.logreg, 27
  - mice.impute.mean, 28
  - mice.impute.norm, 29
  - mice.impute.norm.boot, 30
  - mice.impute.norm.nob, 31
  - mice.impute.norm.predict, 32
  - mice.impute.passive, 33
  - mice.impute.pmm, 34
  - mice.impute.polyreg, 35
  - mice.impute.sample, 36
- \*Topic **datasets**
  - boys, 2
  - nhanes, 45
  - nhanes2, 46
  - popmis, 54
  - windspeed, 65
- \*Topic **hplot**
  - mdc, 18
  - stripplot, 59
- \*Topic **htest**
  - pool, 47
  - pool.compare, 49
  - pool.r.squared, 51
  - pool.scalar, 52
- \*Topic **iteration**
  - mice, 19
  - mice.mids, 37
- \*Topic **manip**
  - cbind.mids, 4
  - complete, 10
  - ibind, 13
  - rbind.mids, 57
- \*Topic **misc**
  - quickpred, 55
  - version, 64
- \*Topic **multivariate**
  - glm.mids, 11
  - lm.mids, 14
  - with.mids, 66
- \*Topic **univar**
  - cc, 6
  - cci, 8
  - ccn, 9
  - md.pairs, 15
  - md.pattern, 17
  - 2L.norm (mice.impute.2L.norm), 24
  - as.mira (mira), 44
  - boys, 2
  - bwplot (stripplot), 59
  - cbind.mids, 4, 14, 59
  - cc, 6, 8, 9
  - cc, data.frame-method (cc), 6
  - cc, matrix-method (cc), 6
  - cc, mids-method (cc), 6
  - cci, 7, 8, 9
  - cci, data.frame-method (cci), 8
  - cci, matrix-method (cci), 8
  - cci, mids-method (cci), 8
  - ccn, 7, 8, 9
  - ccn, data.frame-method (ccn), 9
  - ccn, matrix-method (ccn), 9
  - ccn, mids-method (ccn), 9
  - complete, 10, 24, 38
  - densityplot, 63
  - densityplot (stripplot), 59

- formula, 12, 15, 66
- glm, 12, 28
- glm.fit, 28
- glm.mids, 11, 50
- hcl, 19
- ibind, 6, 13, 59
- ic, 8, 9
- ic(cc), 6
- ic,data.frame-method(cc), 6
- ic,matrix-method(cc), 6
- ic,mids-method(cc), 6
- ici, 7
- ici(cci), 8
- ici,data.frame-method(cci), 8
- ici,matrix-method(cci), 8
- ici,mids-method(cci), 8
- icn(ccn), 9
- icn,data.frame-method(ccn), 9
- icn,matrix-method(ccn), 9
- icn,mids-method(ccn), 9
- is.mids(mids), 39
- is.mipo(mipo), 43
- is.mira(mira), 44
- Lattice, 63
- lda, 27
- lm, 12, 15, 66
- lm.mids, 14, 50
- logreg(mice.impute.logreg), 27
- md.pairs, 15
- md.pattern, 17
- mdc, 18
- mean, 29
- mice, 11, 19, 27–29, 32, 33, 36, 38, 40, 57, 63
- mice.impute.2L.norm, 24
- mice.impute.2L.norm
  - (mice.impute.2L.norm), 24
- mice.impute.lda, 26
- mice.impute.logreg, 27
- mice.impute.mean, 28
- mice.impute.norm, 29, 32
- mice.impute.norm.boot, 30
- mice.impute.norm.nob, 31
- mice.impute.norm.predict, 32
- mice.impute.passive, 33
- mice.impute.pmm, 34
- mice.impute.polr(mice.impute.polyreg),
  - 35
- mice.impute.polyreg, 26, 35
- mice.impute.sample, 36
- mice.mids, 37
- mice.theme(mdc), 18
- mids, 6, 11, 12, 14, 15, 24, 39, 41, 42, 44, 45,
  - 57, 59, 67
- mids-class(mids), 39
- mids2mplus, 40
- mids2spss, 41, 41
- mipo, 40, 43, 45
- mipo-class(mipo), 43
- mira, 12, 15, 40, 44, 44, 67
- mira-class(mira), 44
- multinom, 36
- na.omit, 7, 8
- nhanes, 45, 46
- nhanes2, 45, 46
- norm(mice.impute.norm), 29
- norm.boot(mice.impute.norm.boot), 30
- norm.nob(mice.impute.norm.nob), 31
- norm.predict
  - (mice.impute.norm.predict), 32
- panel.bwplot, 63
- panel.densityplot, 63
- panel.stripplot, 63
- panel.xyplot, 62, 63
- plot(mids), 39
- plot,mids,ANY-method(mids), 39
- plot,mids-method(mids), 39
- plot.mids(mids), 39
- pmm(mice.impute.pmm), 34
- polr, 36
- pool, 43, 44, 47, 52, 53, 67
- pool.compare, 49, 67
- pool.r.squared, 51, 67
- pool.scalar, 52, 52
- popmis, 54
- print, 62
- print,mids-method(mids), 39
- print,mipo-method(mipo), 43
- print,mira-method(mira), 44
- print.trellis, 63
- quickpred, 55

`rbind.mids`, 6, 14, 57

`rgb`, 19

`set.seed`, 24, 38

`stripplot`, 59

`summary`, `mids`-method (`mids`), 39

`summary`, `mipo`-method (`mipo`), 43

`summary`, `mira`-method (`mira`), 44

`trellis.par.set`, 19, 63

`update`, 62

`vcov`, 47, 48, 50

`version`, 64

`windspeed`, 65

`with.mids`, 12, 15, 24, 45, 48, 66

`xyplot`, 19, 61–63

`xyplot` (`stripplot`), 59

`xyplot.mids`, 19