

R documentation

of all in 'mice/man'

July 31, 2000

<code>complete</code>	<i>Produces Imputed Flat Files from Multiply Imputed Data Set (mids)</i>	<code>complete</code>
-----------------------	--	-----------------------

Description

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

Usage

```
data.frame <- complete(x, action=1)
```

Arguments

- `x` An object of class 'mids' (created by the function `mice()`).
- `action` If `action` is a scalar between 1 and `x$m`, the function returns the data with the `action`'s imputation filled in. Thus, `action=1` returns the first completed data set. The can also be one of the following strings: "long", "broad", "repeated". This has the following meaning:
- `action="long"` produces a long matrix with `n*m` rows, containing all imputed data plus two additional variables "_ID_" (containing the row.names) and "_IMP_" (containing the imputation number).
- `action="broad"` produces a broad matrix with `m` times the number of columns in the original data. The first `ncol(x$data)` columns contain the first imputed data matrix. Column names are changed to reflect the imputation number.
- `action="repeated"` produces a broad matrix with `m` times `ncol(x$data)` columns. The first `m` columns give the filled-in first variable. Column names are changed to reflect the imputation number.

Value

A data frame with the imputed values filled in.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

See Also

mice, mids

Examples

```
data(nhanes)
imp <- mice(nhanes)      # do default multiple imputation on a numeric matrix
mat <- complete(imp)    # fills in the first imputation
mat <- complete(imp, 3) # fills in the third imputation
mat <- complete(imp, "long") # produces a long matrix with stacked complete data
mat <- complete(imp, "b") # a broad matrix
cor(mat)                # for numeric mat, produces a blocked correlation matrix, where
                        # each m*m block contains of the same variable pair over different
                        # multiple imputations.
```

glm.mids	<i>Generalized Linear Regression on Multiply Imputed Data</i>	glm.mids
----------	---	----------

Description

Performs repeated glm on a multiply imputed data set

Usage

```
glm.mids(formula=formula(data), family=gaussian, data=sys.parent(), weights,
subset, na.action, start=eta, control=glm.control(...), method="glm.fit",
model=F,x=F, y=T, contrasts=NULL, ...)
```

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.
data	An object of type <code>mids</code> , which stands for 'multiply imputed data set', typically created by function <code>mice()</code> .
family	see <code>glm</code>
weights	
subset	
na.action	
start	
control	
method	
model	
x	
y	
contrasts	

Value

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

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

See Also

`glm`, `mids`, `mira`

Examples

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

```

#
# $analyses[[3]]:
# Call:
# glm(formula = formula, data = data.i)
#
# Coefficients:
# (Intercept)      bmi      chl
# -0.1503616 -0.003002851 0.002130091
#
# Degrees of Freedom: 25 Total; 22 Residual
# Residual Deviance: 3.799126
#
# $analyses[[4]]:
# Call:
# glm(formula = formula, data = data.i)
#
# Coefficients:
# (Intercept)      bmi      chl
# 0.009442083 -0.0237619 0.004631881
#
# Degrees of Freedom: 25 Total; 22 Residual
# Residual Deviance: 3.874522
#
# $analyses[[5]]:
# Call:
# glm(formula = formula, data = data.i)
#
# Coefficients:
# (Intercept)      bmi      chl
# 0.09932161 -0.02168292 0.003857599
#
# Degrees of Freedom: 25 Total; 22 Residual
# Residual Deviance: 4.025066
#
#
# >
#

```

impute.lda	<i>Elementary Imputation Method: Linear Discriminant Analysis</i>	impute.lda
------------	---	------------

Description

Imputes univariate missing data using linear discriminant analysis

Usage

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

Arguments

<code>y</code>	Incomplete data vector of length <code>n</code>
<code>ry</code>	Vector of missing data pattern (F=missing, T=observed)
<code>x</code>	Matrix (<code>n</code> x <code>p</code>) of complete covariates.

Details

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

Value

A vector of length `nmis` with imputations.

Warning

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

Note

This function can be called from within the Gibbs sampler by specifying 'lda' in the `imputationMethod` argument. This method is usually faster and uses less resources than `impute.polyreg`.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

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. (1999). Modern applied statistics with S-Plus (3rd ed). Springer, Berlin.

See Also

`mice`, `lda`, `predict.lda`

`impute.logreg2` *Elementary Imputation Method: Logistic Regression* `impute.logreg2`

Description

Imputes univariate missing data using logistic regression.

Usage

```
imp <- impute.logreg2(y, ry, x)
```

Details

Imputation for binary response variables by the Bayesian logistic regression model. See Rubin (1987, p. 169-170) for a description of the method. The 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.

This method uses direct minimization of the likelihood function by means of V&R function logitreg (V&R, 2nd ed, p. 293).

Value

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

Note

An alternative is `impute.logreg`.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

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`, `logitreg`, `ms`, `impute.logreg`

`impute.logreg` *Elementary Imputation Method: Logistic Regression* `impute.logreg`

Description

Imputes univariate missing data using logistic regression.

Usage

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

Arguments

<code>y</code>	Incomplete data vector of length <code>n</code>
<code>ry</code>	Vector of missing data pattern of length <code>n</code> (F=missing, T=observed)
<code>x</code>	Matrix (<code>n x p</code>) of complete covariates.

Details

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

1. Fit a logit, and find (`bhat`, `V(bhat)`)
2. Draw `BETA` from $N(\text{bhat}, V(\text{bhat}))$
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.

Value

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

Note

An alternative is `impute.logreg2`.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

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.

See Also

`mice`, `glm`, `glm.fit`, `impute.logreg2`

Elementary Imputation Method: Linear Regression Analysis (improper)

Description

Imputes univariate missing data using linear regression analysis (improper version)

Usage

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

Arguments

<code>y</code>	Incomplete data vector of length <code>n</code>
<code>ry</code>	Vector of missing data pattern (F=missing, T=observed)
<code>x</code>	Matrix (<code>n x p</code>) of complete covariates.

Details

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

Value

A vector of length `nmis` with imputations.

Warning

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

Note

This function is provided mainly to allow comparison between proper and improper norm methods.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

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.

See Also

`mice`, `impute.norm`

impute.norm	<i>Elementary Imputation Method: Linear Regression Analysis</i>	impute.norm
-------------	---	-------------

Description

Imputes univariate missing data using linear regression analysis

Usage

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

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (F=missing, T=observed)
x	Matrix (n x p) of complete covariates.

Details

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

Value

A vector of length nmis with imputations.

Note

Using impute.norm for all columns gives results similar to Schafer's norm method (Schafer, 1997), though much slower.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden. 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.

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

impute.passive *Elementary Imputation Method: Passive Imputation* impute.passive

Description

Derive a new variable based on the imputed data

Usage

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

Arguments

<code>data</code>	A data frame
<code>func</code>	A formula specifying the transformations on data

Details

This is a special imputation function for so-called passive imputation. Using this function, 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

<code>t</code>	The tranformed data.
----------------	----------------------

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

See Also

`mice`

<code>impute.pmm</code>	<i>Elementary Imputation Method: Linear Regression Analysis</i>	<code>impute.pmm</code>
-------------------------	---	-------------------------

Description

Imputes univariate missing data using predictive mean matching

Usage

```
impute.pmm(y, ry, x)
```

Arguments

<code>y</code>	Incomplete data vector of length <code>n</code>
<code>ry</code>	Vector of missing data pattern (F=missing, T=observed)
<code>x</code>	Matrix (<code>n x p</code>) of complete covariates.

Details

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

1. Draw β and σ from the proper posterior
2. Compute predicted values for `yobs` and `ymis`
3. For each `ymis`, find the observation with closest predicted value, and take its observed `y` as the imputation.

The matching is on `yhat`, NOT on `y`, which deviates from formula b.

Value

`imp` A vector of length `nmis` with imputations.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

- Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.
- Rubin, D.B. (1987). Multiple imputation for nonresponse in surveys. New York: Wiley.

impute.polyreg *Elementary Imputation Method: Polytomous Regression* impute.polyreg

Description

Imputes missing data in a categorical variable using polytomous regression

Usage

```
impute.polyreg(y, ry, x)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (F=missing, T=observed)
x	Matrix (n x p) of complete covariates.

Details

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.

This algorithm uses the function `multinom` from the libraries `nnet` and `MASS` (Venables and Ripley).

Value

A vector of length `nmis` with imputations.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

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.

See Also

`mice`, `multinom`

impute.sample	<i>Elementary Imputation Method: Simple Random Sample</i>	impute.sample
---------------	---	---------------

Description

Imputes a random sample from the observed y data

Usage

```
impute.sample(y, ry, x=NULL)
```

Arguments

y	Incomplete data vector of length n
ry	Vector of missing data pattern (F=missing, T=observed)
x	Matrix (n x p) of complete covariates.

Details

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

Value

A vector of length nmis with imputations.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

lm.mids	<i>Linear Regression on Multiply Imputed Data</i>	lm.mids
---------	---	---------

Description

Performs repeated linear regression on multiply imputed data set

Usage

```
lm.mids(formula, data, weights, subset, na.action, method="qr", model=F,
         x=F, y=F, contrasts=NULL, ...)
```

Arguments

<code>formula</code>	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.
<code>data</code>	An object of type <code>'mids'</code> , which stands for <code>'multiply imputed data set'</code> , typically created by function <code>mice()</code> .
<code>weights</code>	see <code>lm</code>
<code>subset</code>	see <code>lm</code>
<code>na.action</code>	see <code>lm</code>
<code>method</code>	see <code>lm</code>
<code>model</code>	see <code>lm</code>
<code>x</code>	see <code>lm</code>
<code>y</code>	see <code>lm</code>
<code>contrasts</code>	see <code>lm</code>

Value

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

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

See Also

`lm`, `mids`, `mira`

Examples

```
data(nhanes)
imp <- mice(nhanes)      # do default multiple imputation on a numeric matrix
fit <- lm.mids(bmi~hyp+chl,data=imp)
```

md.pattern

Missing Data Pattern

md.pattern

Description

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

Usage

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

Arguments**Details**

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

Value

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

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

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

See Also

`prelim.norm` (in library 'norm'), `mice`

Examples

```
data(nhanes)
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
```

mice.mids

*Multivariate Imputation by Chained Equations
(Iteration Step)*

mice.mids

Description

Takes a "mids"-object, and produces a new object of class "mids".

Usage

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

Arguments

<code>obj</code>	An object of class "mids", typically produced by a previous call to <code>mice()</code> or <code>mice.mids()</code>
<code>maxit</code>	The number of additional Gibbs sampling iterations.

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.

Value

<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>	A Boolean flag. If TRUE, diagnostic information during the Gibbs sampling iterations will be written to the command window. The default is TRUE.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

See Also**Examples**

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

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

# for example:
#
# > imp$imp$bmi[1,]
#      1      2      3      4      5
```

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

mice

Multivariate Imputation by Chained Equations

mice

Description

Produces an object of class "mids", which stands for 'multiply imputed data set'.

Arguments

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

m Number of multiple imputations. If omitted, m=5 is used.

imputationMethod Can be either a string, or a vector of strings with length ncol(data), specifying the elementary imputation method to be used for each column in data. If specified as a single string, the given 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 `defaultImputationMethod` argument. Columns that need not be imputed have method "". Built-in methods are:

- norm** Bayesian linear regression (Numeric)
- pmm** Predictive mean matching (Numeric)
- mean** Unconditional mean imputation (Numeric)
- logreg** Logistic regression (2 categories)
- logreg2** Logistic regression (direct minimization)(2 categories)
- polyreg** Polytomous logistic regression (≥ 2 categories)
- lda** Linear discriminant analysis (≥ 2 categories)

sample Random sample from the observed values (Any) For example, for the j'th column, the `impute.norm` function that implements the Bayesian linear regression method can be called by specifying the string "norm" as the j'th entry in the vector of strings. The user can write his or her own imputation function, say `impute.myfunc`, and call it for all columns by specifying `imputationMethod="myfunc"`, or for specific columns by specifying `imputationMethod=c("norm","myfunc",...)`.

Special method If the first character of the elementary method is a `~`, then the string is interpreted as the formula argument in a call to `model.frame(formula, data[!r[,j],])`. This provides a simple mechanism for specifying a large variety of dependencies among the variables. For example transformed versions of imputed variables, recodes, interactions, sum scores, and so on, that may themselves be needed in other parts of the algorithm, can be specified in this way. Note that the `~` mechanism works only on those entries which have missing values in the target column. The user should make sure that the combined observed and imputed parts of the target column

make sense. One easy way to create consistency is by coding all entries in the target as `NA`, but for large data sets, this could be inefficient. Moreover, this will not work in S-Plus 4.5. 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.

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

visitSequence A vector of integers of arbitrary length, specifying the column indices of the visiting sequence. The visiting sequence is the column order that is used to impute the data during one iteration of the algorithm. A column may be visited more than once. All incomplete columns that are used as predictors should be visited, or else the function will stop with an error. The default sequence `1:ncol(data)` implies that columns are imputed from left to right.

defaultImputationMethod=c("pmm", "logreg", "polyreg") A vector of three strings containing the default imputation methods for numerical columns, factor columns with 2 levels, and factor columns with more than two levels, respectively. If nothing is specified, the following defaults will be used: *pmm* predictive mean matching (numeric data), *logreg* logistic regression imputation (binary data, factor with 2 levels), *polyreg* polytomous regression imputation categorical data (factor ≥ 2 levels), *maxit* A scalar giving the number of iterations. The default is 5.

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

seed An integer between 0 and 1000 that is used by the `set.seed` function for offsetting the random number generator. The default is 0.

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 appropriate imputation 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 predictor set 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. Users may also write their own imputation functions, and call these from within the algorithm.

In some cases, an imputation model may need transformed data in addition to the original data (e.g. log or quadratic transforms). In order to maintain consistency among different

transformations of the same data, the function has a special built-in method using the `~` mechanism. This method can be used to ensure that a data transform always depends on the most recently generated imputations in the untransformed (active) column.

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.

side effects: Some elementary imputation method require access to the `nnet` or `MASS` libraries of Venables & Ripley. Where needed, these libraries will be attached.

Value

An object of class `mids`, which stands for 'multiply imputed data set'. For a description of the object, see the documentation on "mids".

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. and Oudshoorn, C.G.M.. (1999). Flexible multivariate imputation by MICE. Report PG/VGZ/99.054, TNO Prevention and Health, Leiden.

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

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

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

See Also

`complete`, `mids`, `lm.mids`, `set.seed`

Examples

```
data(nhanes)
imp <- mice(nhanes)      # do default multiple imputation on a numeric matrix
imp
imp$imputations$bmi     # and list the actual imputations
complete(imp)          # show the first completed data matrix
lm.mids(chl~age+bmi+hyp, imp) # repeated linear regression on imputed data

data(nhanes2)
mice(nhanes2,im=c("sample","pmm","logreg","norm")) # imputation on mixed data with a different method
```

mids	<i>Multiply Imputed Data Set</i>	mids
------	----------------------------------	------

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

```
print.mids(object,...)
summary.mids(object,...)
plot.mids(object,...)
```

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.
<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> .
<code>imputationMethod</code>	A vector of strings of length(<code>nvar</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>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>	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.
<code>chainCov</code>	A list with similar structure of <code>itermean</code> , containing the covariances 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 array is only useful for error checking.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

mipo

Multiply Imputed Pooled Analysis
mipo

Description

An object containing the m fit objects of a complete data analysis, plus some additional information. The "mipo" object is generated by the `lm.mids` and `glm.mids` functions. The "mipo" class of objects has methods for the following generic functions: `print`, `summary`.

Usage

```
print.mipo(object,...)
summary.mipo(object,...)
```

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 $m \times \mathit{np\!ar}$ matrix containing the complete data estimates for the $\mathit{np\!ar}$ parameters of the m complete data analyses.
<code>u</code>	An $m \times \mathit{np\!ar} \times \mathit{np\!ar}$ array containing the variance-covariance matrices of the m 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>df</code>	Degrees of freedom associated with the t-statistics.
<code>f</code>	Fractions of missing information.

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

<i>mira</i>	<i>Multiply Imputed Repeated Analysis</i>	<i>mira</i>
-------------	---	-------------

Description

An object containing the m fit objects of a complete data analysis, plus some additional information. The "mira" object is generated by the `lm.mids` and `glm.mids` functions. The "mira" class of objects has methods for the following generic functions: `print`, `summary`.

Usage

```
print.mira(object,...)
summary.mira(object)
```

Value

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

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

References

Van Buuren, S. & Oudshoorn, C.G.M. (2000). Multivariate Imputation by Chained Equations: MICE V1.0 User's manual. Report PG/VGZ/00.038, TNO Prevention and Health, Leiden.

<i>pool</i>	<i>Multiple Imputation Pooling</i>	<i>pool</i>
-------------	------------------------------------	-------------

Description

Pools the results of m repeated complete data analysis

Usage

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

Arguments

<code>object</code>	An object of class 'mira', produced by functions like <code>lm.mids</code> or <code>glm.mids</code> .
<code>dfmethod</code>	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& 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.

R-Specific: The original use of `Varcov` has been removed to `vcov` (VR MASS).

Value

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

Author(s)

Stef van Buuren, Karin Oudshoorn, 2000

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.

Alzola, C.F. and Harrell, F.E. (1999). An introduction to S-Plus and the Hmisc and Design Libraries. <http://hesweb1.med.virginia.edu/biostat/s/index.html>.

See Also

`lm.mids`, `glm.mids`, `vcov`, `print.mira`, `summary.mira`

Examples

```
data(nhanes)
imp <- mice(nhanes)
fit <- lm.mids(bmi~hyp+chl,data=imp)
pool(fit)
# Call: pool(object = fit)
# Pooled coefficients:
# (Intercept)      hyp      chl
# 21.29782 -1.751721 0.04085703
#
# Fraction of information about the coefficients missing due to nonresponse:
# e:
# (Intercept)      hyp      chl
# 0.1592247 0.1738868 0.3117452
#
# > summary(pool(fit))
#               est      se      t      df      Pr(>|t|)
# (Intercept) 21.29781702 4.33668150 4.9110863 16.95890 0.0001329371
#      hyp   -1.75172102 2.30620984 -0.7595671 16.39701 0.4582953905
#      chl    0.04085703 0.02532914 1.6130442 11.50642 0.1338044664
#               lo 95      hi 95 missing      fmi
# (Intercept) 12.14652927 30.4491048      NA 0.1592247
#      hyp   -6.63106456 3.1276225      8 0.1738868
#      chl   -0.01459414 0.0963082     10 0.3117452
```

Index

complete, 1, 19

formula, 2

glm, 2, 3, 7
glm.fit, 7
glm.mids, 2, 21, 23

impute.lda, 4
impute.logreg, 6, 6
impute.logreg2, 5, 7
impute.norm, 8, 9
impute.norm.improper, 8
impute.passive, 10
impute.pmm, 11
impute.polyreg, 5, 12
impute.sample, 13

lda, 5
lm, 2, 14
lm.mids, 13, 19, 21, 23
logitreg, 6

md.pattern, 14
mice, 2, 5–8, 10, 12, 15, 17
mice.mids, 15
mids, 2, 3, 14, 19, 20
mipo, 21
mira, 3, 14, 22
ms, 6
multinom, 12

plot.mids (*mids*), 20
pool, 22
predict.lda, 5
prelim.norm, 15
print.mids (*mids*), 20
print.mipo (*mipo*), 21
print.mira, 23
print.mira (*mira*), 22

set.seed, 19
summary.mids (*mids*), 20
summary.mipo (*mipo*), 21
summary.mira, 23

summary.mira (*mira*), 22
vcov, 23