Package 'norm2'

October 13, 2022

Type Package

Title Analysis of Incomplete Multivariate Data under a Normal Model

Version 2.0.4

Date 2021-02-11

Author Joseph L. Schafer <Joseph.L.Schafer@census.gov>

Maintainer Joseph L. Schafer <Joseph.L.Schafer@census.gov>

Description Functions for parameter estimation, Bayesian posterior simulation and multiple imputation from incomplete multivariate data under a normal model.

Depends R (>= 3.1.0)

Imports stats

License GPL-3

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-02-12 16:50:12 UTC

R topics documented:

norm2-package *Analysis of incomplete multivariate data under a normal model*

Description

Functions for estimation and multiple imputation from incomplete multivariate data under a normal model

Details

The norm2 package provides functions for analyzing incomplete multivariate data using techniques and algorithms described by Schafer (1997). The name of this package derives from the assumed model for the complete data, which is a multivariate normal model. The major functions are:

emNorm	EM algorithm estimating model parameters
mcmcNorm	MCMC algorithm for simulating parameters and missing values
impNorm	Simulate or predict missing values
loglikNorm	Loglikelihood function
logpostNorm	Log-posterior density function
miInference	Combine results from analyses after multiple imputation

The package also includes three datasets:

cholesterol	Cholesterol levels for heart-attack patients
flas	Foreign Language Attitude Scale
marijuana	Changes in heart rate after marijuana use

Note

Fortran source code written by the author for a much earlier version called norm was ported to an R package by Alvaro A. Novo and distributed through the Comprehensive R Archive Network (CRAN). The old package norm is still available on CRAN, but it has some major limitations (e.g., it does not work reliably when the number of variables exceeds 30) and the author does not recommend its use.

Author(s)

Joseph L. Schafer <Joseph.L.Schafer@census.gov>

Maintainer: Joseph L. Schafer <Joseph.L.Schafer@census.gov>

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For more information about functions in norm2, see *User's Guide for* norm2 in the library subdirectory doc.

`cholesterol` *Cholesterol levels for heart-attack patients*

Description

This dataset reports cholesterol levels for 28 patients treated at a Pennsylvania medical center. All patients have cholesterol recorded on day 2 and day 4 after attack, but some have missing values on day 14. These data were analyzed by Schafer (1997, Chap. 5).

Usage

```
data(cholesterol)
```

Format

a data frame with 28 rows and 3 variables:

`Y1`	cholesterol 2 days after heart attack.
`Y2`	cholesterol 4 days after heart attack.
`Y3`	cholesterol 14 days after heart attack.

Source

Ryan, B.F. and Joiner, B.L. (1994) *Minitab Handbook* (Third edition). Belmont, CA: Wadsworth.

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For example analyses of this dataset using functions in the `norm2` package, see the manual *NORM Package for R, Version 2* in the library subdirectory `doc`.

`emNorm` *EM algorithm for incomplete multivariate normal data*

Description

Computes maximum likelihood estimates and posterior modes from incomplete multivariate data under a normal model.

Usage

```
emNorm(obj, ...)


## Default S3 method:
emNorm(obj, x = NULL, intercept = TRUE,
   iter.max = 1000, criterion = NULL, estimate.worst = TRUE,
   prior = "uniform", prior.df = NULL, prior.sscp = NULL,
   starting.values = NULL, ...)


## S3 method for class 'formula'
emNorm(formula, data, iter.max = 1000,
   criterion = NULL, estimate.worst = TRUE, prior = "uniform",
   prior.df = NULL, prior.sscp = NULL, starting.values = NULL, ...)


## S3 method for class 'norm'
emNorm(obj, iter.max = 1000,
   criterion = obj$criterion, estimate.worst = obj$estimate.worst,
   prior = obj$prior, prior.df = obj$prior.df,
   prior.sscp = obj$prior.sscp, starting.values = obj$param, ...)
```

Arguments

`obj`	an object used to select a method. It may be `y`, a numeric matrix, vector or data frame containing response variables to be modeled as multivariate normal. Missing values (`NA`s) are allowed. If `y` is a data frame, any factors or ordered factors will be replaced by their internal codes, and a warning will be given. Alternatively, this first argument may be a `formula` as described below, or an object of class `"norm"` resulting from a call to `emNorm` or `mcmcNorm`; see DETAILS.
`x`	a numeric matrix, vector or data frame of covariates to be used as model predictors. Missing values (`NA`'s) are not allowed. If `x` is a matrix, it must have the same number of rows as `y`. If `x` is a data frame, any factors or ordered factors will be replaced by their internal codes, and a warning is given. If `NULL`, it defaults to `x = rep(1,nrow(y))`, an intercept-only model.
`intercept`	if `TRUE`, then a column of `1`'s is appended to `x`. Ignored if `x = NULL`.
`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model which is provided in lieu of `y` and `x`. The details of model specification are given under DETAILS.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `emNorm` is called.

`iter.max`	maximum number of iterations to be performed. Each iteration consists of an Expectation or E-step followed by a Maximization or M-step. The procedure halts if it has not converged by this many iterations.
`criterion`	convergence criterion. The procedure halts if the maximum relative change in all parameters from one iteration to the next falls below this value. If `NULL`, then the default criterion of `1e-05` is used.
`estimate.worst`	if `TRUE`, then upon convergence of the EM algorithm, a procedure is attempted to numerically estimate the worst fraction of missing information and the worst linear function of the parameters; see DETAILS.
`prior`	should be `"uniform"`, `"jeffreys"`, `"ridge"` or `"invwish"`. If `"ridge"` then `prior.df` must be supplied. If `"invwish"` then `prior.df` and `prior.sscp` must be supplied. For more information, see DETAILS.
`prior.df`	prior degrees of freedom for a ridge (`prior="ridge"`) or inverted Wishart (`prior="invwish"`) prior.
`prior.sscp`	prior sums of squares and cross-products (SSCP) matrix for an inverted Wishart prior (`prior="invwish"`).
`starting.values`	optional starting values for the model parameters. This must be a list with two named components, `beta` and `sigma`, which are numeric matrices with correct dimensions; see DETAILS.
`...`	values to be passed to the methods.

Details

There are three different ways to specify the data and model when calling `emNorm`:

- by directly supplying as the initial argument a matrix of numeric response variables `y`, along with an optional matrix of predictor variables `x`;
- by supplying a model specification `formula`, along with an optional data frame `data`; or
- by supplying an object of class `"norm"`, which was produced by an earlier call to `emNorm` or `mcmcNorm`.

In the first case, the matrix `y` is assumed to have a multivariate normal linear regression on `x` with coefficients `beta` and covariance matrix `sigma`, where `dim(beta)=c(ncol(x),ncol(y))` and `dim(sigma)=c(ncol(y),ncol(y))`. Missing values `NA` are allowed in `y` but not in `x`.

In the second case, `formula` is a formula for a (typically multivariate) linear regression model in the manner expected by `lm`. A formula is given as `y ~ model`, where `y` is either a single numeric variable or a matrix of numeric variables bound together with the function `cbind`. The right-hand side of the formula (everything to the right of `~`) is a linear predictor, a series of terms separated by operators `+`, `:` or `*` to specify main effects and interactions. Factors are allowed on the right-hand side and will enter the model as contrasts among the `levels`. The intercept term `1` is included by default; to remove the intercept, use `-1`.

In the third case, the initial argument to `emNorm` is an object of class `"norm"` returned by a previous call to `emNorm` or `mcmcNorm`. The value of the parameters carried in this object (the estimates from the last iteration of EM or the simulated values from the last iteration of MCMC) will be used as the starting values. This can be useful, for example, if a previous run of EM did not converge by max.iter iterations. Supplying the result from that EM run as the sole argument to emNorm allows the algorithm to continue from where it was halted.

If prior="uniform" (the default), the EM algorithm computes a maximum-likelihood estimate for the parameters beta and sigma; otherwise it computes a posterior mode.

If prior="invwish" then an inverted Wishart prior distribution is applied to sigma with hyperparameters prior.df (a scalar) and prior.sscp (a symmetric, positive definite matrix of the same dimension as sigma). Using the device of imaginary results, we can interpret prior.sscp/prior.df as a prior guess for sigma, and prior.df as the prior degrees of freedom on which this guess is based.

The usual noninformative prior for the normal regression model (prior="jeffreys") is equivalent to the inverted Wishart density with prior.df equal to 0 and prior.sscp equal to a matrix of 0's.

The ridge prior (prior="ridge") is a special case of the inverted Wishart (Schafer, 1997). The prior guess for sigma is a diagonal matrix with diagonal elements estimated by regressing the observed values in each column of y on the corresponding rows of x. When prior="ridge", the user must supply a value for prior.df, which determines how strongly the estimated correlations are smoothed toward zero.

If estimate.worst, then upon convergence, a procedure is run to numerically estimate the worst fraction of missing information and the worst linear function of the parameters. The worst fraction of missing information is closely related to EM's convergence rate. Values near one correspond to slow convergence, and values near zero indicate fast convergence. If there are no missing values in the response variables, the worst fraction of missing information is exactly zero, and EM converges after one step from any starting values. The worst linear function is a linear combination of the parameters (elements of beta and sigma) for which the rate of missing information is highest.

For details of the EM algorithm, see the manual distributed with the norm2 package in the library subdirectory doc.

Value

a list whose class attribute has been set to "norm". This object may be passed as the first argument in subsequent calls to emNorm, mcmcNorm, impNorm, loglikNorm or logpostNorm. The object also carries the original data and specifies the prior distribution, so that these do not need to be provided again.

To see a summary of this object, use the generic function summary, which passes the object to summary.norm.

Components of the list may also be directly accessed and examined by the user. Components which may be of interest include:

iter	number of EM iterations actually performed.
rel.diff	maximum relative difference between the parameters the last two iterations.
converged	logical value indicating whether the algorithm converged by iter iterations. Will be TRUE if rel.diff<=criterion.
loglik	a numeric vector of length iter reporting the logarithm of the observed-data likelihood function at the start of each iteration. If prior="uniform" then the loglikelihood values will be non-decreasing.

logpost	a numeric vector of length iter reporting the logarithm of the observed-data posterior density function at the start of each iteration. The log-posterior density values will be non-decreasing. If prior="uniform" then the log-posterior density and loglikelihood will be identical.
param	a list with elements beta and sigma containing the estimated parameters after the final iteration of EM. This may be supplied as starting values to emNorm or mcmcNorm, or as an argument to impNorm, loglikNorm or logpostNorm .
miss.patt	logical matrix with ncol(y) columns reporting the missingness patterns seen in y. Each row of miss.patt corresponds to a distinct missingness pattern, with TRUE indicating that the y-variable is missing and FALSE indicating that the y-variable is observed.
miss.patt.freq	integer vector of length nrow(miss.patt) indicating, for each missingness pattern, the number of cases or rows of y having that pattern.
which.patt	integer vector of length nrow(y) indicating the missingness pattern for each row of y. Thus is.na(y[i,]) is the same thing as miss.patt[which.patt[i],].

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For more information about this function and other functions in norm2, see the manual *NORM Package for R, Version 2* in the library subdirectory doc.

See Also

mcmcNorm, summary.norm, impNorm, loglikNorm, logpostNorm

Examples

```
## run EM for marijuana data with strict convergence criterion
data(marijuana)
result <- emNorm(marijuana, criterion=1e-06)

## re-run with ridge prior and examine results
result <- emNorm(marijuana, prior="ridge", prior.df=0.5)
summary(result)
```

`flas` *Foreign Language Attitude Scale*

Description

This dataset comes from a study to evaluate the reliability of the Foreign Lanuage Attitude Scale, an instrument for predicting success in the study or foreign languages (Raymond and Roberts, 1983). The questionnaire was given to 279 students enrolled in four different language courses (French, German, Spanish, Russian) at Penn State University. This dataset includes FLAS score, final grade in the course, and other variables for predicting achievement. These data were analyzed by Schafer (1997, Chap. 6).

Usage

```
data(flas)
```

Format

a data frame with 279 rows and 14 variables:

`LAN2` 1=Spanish, 0=other.

`LAN3` 1=German, 0=other.

`LAN4` 1=Russian, 0=other.

`AGE` age group (1=less than 20, 2=20+).

`PRI` number of prior foreign language courses (1=none, 2=1-2, 3=3+).

`SEX` 0=male, 1=female

`MLAT` Modern Language Aptitude Test

`FLAS` Foreign Language Attitude Scale

`SATV` Scholastic Aptitude Test, verbal score

`SATM` Scholastic Aptitude Test, math score

`ENG` score on Penn State English placement exam

`HGPA` high school grade point average

`CGPA` current college grade point average

`GRD` final grade in foreign language course (1=B or lower, 2=A)

Source

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

References

Raymond, M.R. and Roberts, D.M. (1983) Development and validation of a foreign language attitude scale. *Educational and Psychological Measurement*, 43, 1239-1246.

For example analyses of this dataset using functions in the norm2 package, see the manual *NORM Package for R, Version 2* in the library subdirectory doc.

impNorm *Imputation and prediction for incomplete multivariate normal data*

Description

Simulates or predicts missing values from their predictive distribution given the observed data under a normal model with fixed parameters.

Usage

```
impNorm(obj, ...)


## Default S3 method:
impNorm(obj, x = NULL, intercept = TRUE, param,
   seeds = NULL, method = "random", ...)


## S3 method for class 'formula'
impNorm(formula, data, param,
   seeds = NULL, method = "random", ...)


## S3 method for class 'norm'
impNorm(obj, param = obj$param, seeds = NULL,
   method = "random", ...)
```

Arguments

obj	an object used to select a method. It may be y, a numeric matrix, vector or data frame of responses to be modeled as normal. Missing values (NAs) are allowed. If y is a data frame, any factors or ordered factors will be replaced by their internal codes, and a warning will be given. Alternatively, this first argument may be an object of class "norm" resulting from a call to emNorm or mcmcNorm; see DETAILS.

`x`	a numeric matrix, vector or data frame of covariates to be used as predictors for `y`. Missing values (`NA`'s) are not allowed. If `x` is a matrix, it must have the same number of rows as `y`. If `x` is a data frame, any factors or ordered factors are replaced by their internal codes, and a warning is given. If `NULL`, it defaults to `x = rep(1,nrow(y))`, an intercept-only model.
`intercept`	if `TRUE`, then a column of `1`'s is appended to `x`. Ignored if `x = NULL`.
`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model which is provided in lieu of `y` and `x`. The details of model specification are given under DETAILS.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `impNorm` is called.
`param`	assumed values for the model parameters. This must be a list with two named components, `beta` and `sigma`, which are numeric matrices with correct dimensions. In most circumstances, the parameter values will be obtained from a run of `emNorm` or `mcmcNorm`; see DETAILS.
`seeds`	two integers to initialize the random number generator; see DETAILS.
`method`	if `"random"`, the missing values in each row of `y` will be simulated from their joint predictive distribution given `x` and the observed values in `y`. If `"predict"`, missing values will be replaced by regression predictions given the observed values. See DETAILS.
`...`	values to be passed to the methods.

Details

This function is used primarily in conjunction with `mcmcNorm` to draw multiple imputations by the multiple-chain method. In those instances, the simplest way to call `impNorm` is to provide an object of class `"norm"` as its first argument, where that object is the result of a call to `mcmcNorm`. The parameter values stored in that object will then be passed to `impNorm` automatically.

Alternatively, one may call `impNorm` by providing as the first argument `y`, a vector or matrix of data to be modeled as normal, and an optional vector or matrix of predictors `x`. Missing values `NA` are allowed in `y` but not in `x`.

A third way to call `impNorm` is to provide `formula`, a formula for a (typically multivariate) linear regression model in the manner expected by `lm`. A formula is given as `y ~ model`, where `y` is either a single numeric variable or a matrix of numeric variables bound together with the function `cbind`. The right-hand side of the formula (everything to the right of `~`) is a linear predictor, a series of terms separated by operators `+`, `:` or `*` to specify main effects and interactions. Factors are allowed on the right-hand side and will enter the model as contrasts among the `levels`. The intercept term `1` is included by default; to remove the intercept, use `-1`.

`norm2` functions use their own internal random number generator which is seeded by two integers, for example, `seeds=c(123,456)`, which allows results to be reproduced in the future. If `seeds=NULL` then the function will seed itself with two random integers from R. Therefore, results can also be made reproducible by calling `set.seed` beforehand and taking `seeds=NULL`.

Value

a data matrix resembling the original data `y`, but with `NA`'s replaced by simulated values or predictions.

Author(s)

Joe Schafer <`Joseph.L.Schafer@census.gov`>

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For more information about this function and other functions in the `norm2` package, see *User's Guide for* `norm2` in the library subdirectory `doc`.

See Also

`emNorm`, `mcmcNorm`

Examples

```
## run EM for marijuana data with ridge prior
data(marijuana)
emResult <- emNorm(marijuana, prior="ridge", prior.df=0.5)

## generate 25 multiple imputations by running 25 chains
## of 100 iterations each, starting each chain at the
## posterior mode
set.seed(456)
imp.list <- as.list(NULL)
for(m in 1:25){
   mcmcResult <- mcmcNorm(emResult, iter=100)
   imp.list[[m]] <- impNorm(mcmcResult)}
```

`loglikNorm` *Observed-data loglikelikehood for incomplete multivariate normal data*

Description

Computes the observed-data loglikelihood function at given parameter values for an incomplete dataset under a normal model.

Usage

```
loglikNorm(obj, ...)

## Default S3 method:
loglikNorm(obj, x = NULL, intercept = TRUE, param, ...)

## S3 method for class 'formula'
loglikNorm(formula, data, param, ...)

## S3 method for class 'norm'
loglikNorm(obj, param = obj$param, ...)
```

Arguments

`obj`	an object used to select a method. It may be `y`, a numeric matrix, vector or data frame of responses to be modeled as normal. Missing values (`NA`s) are allowed. If `y` is a data frame, any factors or ordered factors will be replaced by their internal codes, and a warning will be given. Alternatively, this first argument may be a `formula` as described below, or an object of class `"norm"` resulting from a call to `emNorm` or `mcmcNorm`; see DETAILS.
`x`	a numeric matrix, vector or data frame of covariates to be used as predictors for `y`. Missing values (`NA`'s) are not allowed. If `x` is a matrix, it must have the same number of rows as `y`. If `x` is a data frame, any factors or ordered factors are replaced by their internal codes, and a warning is given. If `NULL`, it defaults to `x = rep(1,nrow(y))`, an intercept-only model.
`intercept`	if `TRUE`, then a column of `1`'s is appended to `x`. Ignored if `x = NULL`.
`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model which is provided in lieu of `y` and `x`. The details of model specification are given under DETAILS.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `loglikNorm` is called.
`param`	assumed values for the model parameters. This must be a list with two named components, `beta` and `sigma`, which are numeric matrices with correct dimensions. In most circumstances, the parameter values will be obtained from a run of `emNorm` or `mcmcNorm`; see DETAILS.
`...`	values to be passed to the methods.

Details

The simplest way to call `loglikNorm` is to provide an object of class `"norm"` as its sole argument, where that object is the result of a call to `emNorm` or `mcmcNorm`. The parameter values stored in that object will then be passed to `loglikNorm` automatically.

Alternatively, one may call `loglikNorm` by providing as the first argument `y`, a vector or matrix of data to be modeled as normal, and an optional vector or matrix of predictors `x`. Missing values `NA` are allowed in `y` but not in `x`.

A third way to call `loglikNorm` is to provide `formula`, a formula for a (typically multivariate) linear regression model in the manner expected by `lm`. A formula is given as `y ~ model`, where `y` is either a single numeric variable or a matrix of numeric variables bound together with the function `cbind`. The right-hand side of the formula (everything to the right of `~`) is a linear predictor, a series of terms separated by operators `+`, `:` or `*` to specify main effects and interactions. Factors are allowed on the right-hand side and will enter the model as contrasts among the `levels`. The intercept term `1` is included by default; to remove the intercept, use `-1`.

Calling `loglikNorm` is equivalent to calling `logpostNorm` with `prior="uniform"`.

Value

a numeric value reporting the observed-data loglikelihood

Author(s)

Joe Schafer `<Joseph.L.Schafer@census.gov>`

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For more information about this function and other functions in the `norm2` package, see *User's Guide for* `norm2` in the library subdirectory `doc`.

See Also

`emNorm`, `mcmcNorm`, `logpostNorm`

Examples

```
## run EM for cholesterol data and display the
## loglikelihood values at all iterations
data(cholesterol)
emResult <- emNorm(cholesterol)
print( emResult$loglik )

## compute the loglikelihood at the final estimate
## and compare it to the last loglikelihood value
## reported by emNorm
loglik.max <- loglikNorm(emResult)
```

```
print( loglik.max - emResult$loglik[ emResult$iter ] )

## The result from loglikNorm is slightly higher,
## because the last value reported by emNorm is the
## loglikelihood at the BEGINNING of the last iteration
```

---

logpostNorm — *Observed-data log-posterior density for incomplete multivariate normal data*

Description

Computes the observed-data log-posterior density function at given parameter values for an incomplete dataset under a normal model.

Usage

```
logpostNorm(obj, ...)


## Default S3 method:
logpostNorm(obj, x = NULL, intercept = TRUE, param,
   prior = "uniform", prior.df = NULL, prior.sscp = NULL, ...)


## S3 method for class 'formula'
logpostNorm(formula, data, param,
   prior = "uniform", prior.df = NULL, prior.sscp = NULL, ...)


## S3 method for class 'norm'
logpostNorm(obj, param = obj$param, prior = obj$prior,
   prior.df = obj$prior.df, prior.sscp = obj$prior.sscp, ...)
```

Arguments

`obj`	an object used to select a method. It may be `y`, a numeric matrix, vector or data frame of responses to be modeled as normal. Missing values (`NA`s) are allowed. If `y` is a data frame, any factors or ordered factors will be replaced by their internal codes, and a warning will be given. Alternatively, this first argument may be a `formula` as described below, or an object of class `"norm"` resulting from a call to `emNorm` or `mcmcNorm`; see DETAILS.
`x`	a numeric matrix, vector or data frame of covariates to be used as predictors for `y`. Missing values (`NA`'s) are not allowed. If `x` is a matrix, it must have the same

	number of rows as `y`. If `x` is a data frame, any factors or ordered factors are replaced by their internal codes, and a warning is given. If `NULL`, it defaults to `x = rep(1,nrow(y))`, an intercept-only model.
`intercept`	if `TRUE`, then a column of `1`'s is appended to `x`. Ignored if `x = NULL`.
`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model which is provided in lieu of `y` and `x`. The details of model specification are given under DETAILS.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `logpostNorm` is called.
`param`	assumed values for the model parameters. This must be a list with two named components, `beta` and `sigma`, which are numeric matrices with correct dimensions. In most circumstances, the parameter values will be obtained from a run of `emNorm` or `mcmcNorm`; see DETAILS.
`prior`	should be `"uniform"`, `"jeffreys"`, `"ridge"` or `"invwish"`. If `"ridge"` then `prior.df` must be supplied. If `"invwish"` then `prior.df` and `prior.sscp` must be supplied.
`prior.df`	prior degrees of freedom for a ridge (`prior="ridge"`) or inverted Wishart (`prior="invwish"`) prior.
`prior.sscp`	prior sums of squares and cross-products (SSCP) matrix for an inverted Wishart prior (`prior="invwish"`).
`...`	values to be passed to the methods.

Details

The simplest way to call `logpostNorm` is to provide an object of class `"norm"` as its sole argument, where that object is the result of a call to `emNorm` or `mcmcNorm`. The parameter values stored in that object will then be passed to `logpostNorm` automatically.

Alternatively, one may call `logpostNorm` by providing as the first argument `y`, a vector or matrix of data to be modeled as normal, and an optional vector or matrix of predictors `x`. Missing values `NA` are allowed in `y` but not in `x`.

A third way to call `logpostNorm` is to provide `formula`, a formula for a (typically multivariate) linear regression model in the manner expected by `lm`. A formula is given as `y ~ model`, where `y` is either a single numeric variable or a matrix of numeric variables bound together with the function `cbind`. The right-hand side of the formula (everything to the right of `~`) is a linear predictor, a series of terms separated by operators `+`, `:` or `*` to specify main effects and interactions. Factors are allowed on the right-hand side and will enter the model as contrasts among the `levels`. The intercept term `1` is included by default; to remove the intercept, use `-1`.

Value

a numeric value reporting the observed-data log-posterior density

Author(s)

Joe Schafer `<Joseph.L.Schafer@census.gov>`

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For more information about this function and other functions in the norm2 package, see *User's Guide for* norm2 in the library subdirectory doc.

See Also

emNorm, mcmcNorm, loglikNorm

Examples

```
## run EM for marijuana data with ridge prior and print the
## last value of the log-posterior density
data(marijuana)
emResult <- emNorm(marijuana, prior="ridge", prior.df=0.5)
print( emResult$logpost[ emResult$iter ] )

## compute the log-posterior density at the final estimate
## and compare it to the last value reported by emNorm
logpost.max <- logpostNorm(emResult)
print( logpost.max - emResult$logpost[ emResult$iter ] )

## The result from logpostNorm is slightly higher,
## because the last value reported by emNorm is the
## log-posterior at the BEGINNING of the last iteration
```

---

marijuana *Changes in heart rate after marijuana use*

Description

This dataset comes from a pilot study to investigate the effects of marijuana. Nine male subjects received three treatments each (placebo, low dose and high dose) and the order of treatments was balanced in a replicated 3 x 3 Latin square. Under each treatment, the change in heart rate (beats per minute above baseline) was recorded 15 minutes after smoking and 90 minutes after smoking, producing six measures per subject, but five of the 54 data values are missing. These data were analyzed by Schafer (1997, Chap. 5).

Usage

```
data(marijuana)
```

Format

a data frame with 9 rows and 6 variables.

Source

Weil, A.T., Zinberg, N.E. and Nelson, J.M. (1968) Clinical and psychological effects of marihuana in man. *Science*, 62, 1234-1242.

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data.* London: Chapman & Hall/CRC Press.

For example analyses of this dataset using functions in the `norm2` package, see the manual *NORM Package for R, Version 2* in the library subdirectory `doc`.

`mcmcNorm` *MCMC algorithm for incomplete multivariate normal data*

Description

Simulates parameters and missing values from a joint posterior distribution under a normal model using Markov chain Monte Carlo.

Usage

```
mcmcNorm(obj, ...)


## Default S3 method:
mcmcNorm(obj, x = NULL, intercept = TRUE,
   starting.values, iter = 1000, multicycle = NULL,
   seeds = NULL, prior = "uniform",
   prior.df = NULL, prior.sscp = NULL, save.all.series = TRUE,
   save.worst.series = FALSE, worst.linear.coef = NULL,
   impute.every = NULL, ...)


## S3 method for class 'formula'
mcmcNorm(formula, data, starting.values,
   iter = 1000, multicycle = NULL, seeds = NULL, prior = "uniform",
   prior.df = NULL, prior.sscp = NULL, save.all.series = TRUE,
   save.worst.series = FALSE, worst.linear.coef = NULL,
   impute.every=NULL, ...)


## S3 method for class 'norm'
mcmcNorm(obj, starting.values = obj$param,
   iter = 1000, multicycle = obj$multicycle,
   seeds = NULL, prior = obj$prior, prior.df = obj$prior.df,
   prior.sscp = obj$prior.sscp,
   save.all.series = !(obj$method=="MCMC" & is.null( obj$series.beta )),
   save.worst.series = !is.null( obj$worst.linear.coef ),
   worst.linear.coef = obj$worst.linear.coef,
   impute.every = obj$impute.every, ...)
```

Arguments

`obj`	an object used to select a method. It may be `y`, a numeric matrix, vector or data frame containing response variables to be modeled as normal. Missing values (`NA`s) are allowed. If `y` is a data frame, any factors or ordered factors will be replaced by their internal codes, and a warning will be given. Alternatively, this first argument may be a `formula` as described below, or an object of class `"norm"` resulting from a call to `emNorm` or `mcmcNorm`; see DETAILS.
`x`	a numeric matrix, vector or data frame of covariates to be used as predictors for `y`. Missing values (`NA`'s) are not allowed. If `x` is a matrix, it must have the same number of rows as `y`. If `x` is a data frame, any factors or ordered factors are replaced by their internal codes, and a warning is given. If `NULL`, it defaults to `x = rep(1,nrow(y))`, an intercept-only model.
`intercept`	if `TRUE`, then a column of `1`'s is appended to `x`. Ignored if `x = NULL`.
`formula`	an object of class `"formula"` (or one that can be coerced to that class): a symbolic description of the model which is provided in lieu of `y` and `x`. The details of model specification are given under DETAILS.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `mcmcNorm` is called.
`starting.values`	starting values for the model parameters. This must be a list with two named components, `beta` and `sigma`, which are numeric matrices with correct dimensions. In most circumstances, the starting values will be obtained from a prior run of `emNorm` or `mcmcNorm`; see DETAILS.
`iter`	number of iterations to be performed. By default, each iteration consists of one Imputation or I-step followed by one Posterior or P-step, but this can be changed by `multicycle`.
`multicycle`	number of cycles per iteration, with `NULL` equivalent to `multicycle=1`. Specifying `multicycle=`*k* for some *k*>1 instructs `mcmcNorm` to perform the I-step and P-step cycle `k` times within each iteration; see DETAILS.
`seeds`	two integers to initialize the random number generator; see DETAILS.
`prior`	should be `"uniform"`, `"jeffreys"`, `"ridge"` or `"invwish"`. If `"ridge"` then `prior.df` must be supplied. If `"invwish"` then `prior.df` and `prior.sscp` must be supplied. For more information, see DETAILS.
`prior.df`	prior degrees of freedom for a ridge (`prior="ridge"`) or inverted Wishart (`prior="invwish"`) prior.

`prior.sscp`	prior sums of squares and cross-products (SSCP) matrix for an inverted Wishart prior (`prior="invwish"`).
`save.all.series`	if `TRUE`, then the simulated values of all parameters at all iterations will be saved.
`save.worst.series`	if `TRUE`, then the simulated values of the worst linear function of the parameters will be saved. Under ordinary circumstances, this function will have been estimated by `emNorm` after the EM algorithm converged.
`worst.linear.coef`	vector or coefficients that define the worst linear function of the parameters. Under ordinary circumstances, these are provided automatically in the result from `emNorm`.
`impute.every`	how many iterations to perform between imputations? If `impute.every=`*k*, then the simulated values for the missing data after every *k* iterations will be saved, resulting in `floor(iter/impute.every)` multiple imputations. If `NULL`, then no imputations will be saved.
`...`	values to be passed to the methods.

Details

There are three different ways to specify the data and model when calling `mcmcNorm`:

- by directly supplying as the initial argument a matrix of numeric response variables `y`, along with an optional matrix of predictor variables `x`;
- by supplying a model specification `formula`, along with an optional data frame `data`; or
- by supplying an object of class `"norm"`, which was produced by an earlier call to `emNorm` or `mcmcNorm`.

In the first case, the matrix `y` is assumed to have a multivariate normal linear regression on `x` with coefficients `beta` and covariance matrix `sigma`, where `dim(beta)=c(ncol(x),ncol(y))` and `dim(sigma)=c(ncol(y),ncol(y))`. Missing values `NA` are allowed in `y` but not in `x`.

In the second case, `formula` is a formula for a (typically multivariate) linear regression model in the manner expected by `lm`. A formula is given as `y ~ model`, where `y` is either a single numeric variable or a matrix of numeric variables bound together with the function `cbind`. The right-hand side of the formula (everything to the right of `~`) is a linear predictor, a series of terms separated by operators `+`, `:` or `*` to specify main effects and interactions. Factors are allowed on the right-hand side and will enter the model as contrasts among the `levels`. The intercept term `1` is included by default; to remove the intercept, use `-1`.

In the third case, the initial argument to `mcmcNorm` is an object of class `"norm"` returned by a previous call to `emNorm` or `mcmcNorm`. The value of the parameters carried in this object (the estimates from the last iteration of EM or the simulated values from the last iteration of MCMC) will be used as the starting values.

The matrix `y` is assumed to have a multivariate normal linear regression on `x` with coefficients `beta` and covariance matrix `sigma`, where `dim(beta)=c(ncol(x),ncol(y))` and `dim(sigma)=c(ncol(y),ncol(y))`.

Starting values for the parameters must be provided. In most cases these will be the result of a previous call to `emNorm` or `mcmcNorm`. If the starting values are close to the mode (i.e., if they are the result of an EM run that converged) then the worst linear function of the parameters will be saved at each iteration. If the starting values are the result of a previous run of MCMC, then the new run will be a continuation of the same Markov chain.

If `multicycle=`*k* for some *k*>1, then the length of the saved parameter series will be reduced by a factor of *k*, and the serial correlation in the series will also be reduced. This option is useful in large problems with many parameters and in slowly converging problems for which many iterations are needed.

`norm2` functions use their own internal random number generator which is seeded by two integers, for example, `seeds=c(123,456)`, which allows results to be reproduced in the future. If `seeds=NULL` then the function will seed itself with two random integers from R. Therefore, results can also be made reproducible by calling `set.seed` beforehand and taking `seeds=NULL`.

If `prior="invwish"` then an inverted Wishart prior distribution is applied to `sigma` with hyperparameters `prior.df` (a scalar) and `prior.sscp` (a symmetric, positive definite matrix of the same dimension as `sigma`). Using the device of imaginary results, we can interpret `prior.sscp/prior.df` as a prior guess for `sigma`, and `prior.df` as the prior degrees of freedom on which this guess is based.

The usual noninformative prior for the normal regression model (`prior="jeffreys"`) is equivalent to the inverted Wishart density with `prior.df` equal to 0 and `prior.sscp` equal to a matrix of 0's.

The ridge prior (`prior="ridge"`) is a special case of the inverted Wishart (Schafer, 1997). The prior guess for `sigma` is a diagonal matrix with diagonal elements estimated by regressing the observed values in each column of `y` on the corresponding rows of `x`. When `prior="ridge"`, the user must supply a value for `prior.df`, which determines how strongly the estimated correlations are smoothed toward zero.

If the first argument to `mcmcNorm` is an object of class `"norm"`, then the parameter values stored in that object will automatically be used as starting values.

For details of the MCMC algorithm, see the manual distributed with the NORM package in the subdirectory `doc`.

Value

a list whose `class` attribute has been set to `"norm"`. This object may be passed as the first argument in subsequent calls to `emNorm`, `mcmcNorm`, `impNorm`, `loglikNorm` or `logpostNorm`. The object also carries the original data and specifies the prior distribution, so that these do not need to be provided again.

To see a summary of this object, use the generic function `summary`, which passes the object to `summary.norm`.

Components of the list may also be directly accessed and examined by the user. Components which may be of interest include:

`iter`	number of MCMC iterations performed.
`param`	a list with elements `beta` and `sigma` containing the estimated parameters after the final iteration of MCMC. This may be supplied as starting values to `emNorm` or `mcmcNorm`, or as an argument to `impNorm`, `loglikNorm` or `logpostNorm`.

`loglik`	a numeric vector of length `iter` reporting the logarithm of the observed-data likelihood function at the start of each iteration.
`logpost`	a numeric vector of length `iter` reporting the logarithm of the observed-data posterior density function at the start of each iteration.
`series.worst`	a time-series object (class `"ts"`) which contains the simulated values of the worst linear function of the parameters from all iterations. This will be present if the starting values provided to `mcmcNorm` were close enough to the mode to provide a reliable estimate of the worst linear function. The dependence in this series tends to be higher than for other parameters, so examining the dependence by plotting the series with `plot` or its autocorrelation function with `acf` may help the user to judge how quickly the Markov chain achieves stationarity. For the definition of the worst linear function, see the manual accompanying the NORM package in the subdirectory `doc`.
`series.beta`	a multivariate time-series object (class `"ts"`) which contains the simulated values of the coefficients `beta` from all iterations. This will present if `save.all.series=TRUE`.
`series.sigma`	a multivariate time-series object (class `"ts"`) which contains the simulated values of the variances and covariances (elements of the lower triangle of `sigma`) from all iterations. This will be present if `save.all.series=TRUE`.
`imp.list`	a list containing the multiple imputations. Each component of this list is a data matrix resembling `y`, but with `NA`'s replaced by imputed values. The length of the list depends on the values of `iter` and `impute.every`.
`miss.patt`	logical matrix with `ncol(y)` columns reporting the missingness patterns seen in `y`. Each row of `miss.patt` corresponds to a distinct missingness pattern, with `TRUE` indicating that the variable is missing and `FALSE` indicating that the variable is observed.
`miss.patt.freq`	integer vector of length `nrow(miss.patt)` indicating, for each missingness pattern, the number of cases or rows of `y` having that pattern.
`which.patt`	integer vector of length `nrow(y)` indicating the missingness pattern for each row of `y`. Thus `is.na( y[i,] )` is the same thing as `miss.patt[ which.patt[i], ]`.

Author(s)

Joe Schafer <`Joseph.L.Schafer@census.gov`>

References

Schafer, J.L. (1997) *Analysis of Incomplete Multivariate Data*. London: Chapman & Hall/CRC Press.

For more information about this function and other functions in the NORM package, see *User's Guide for* `norm2` in the library subdirectory `doc`.

See Also

`emNorm`, `summary.norm`, `impNorm`, `loglikNorm`, `logpostNorm`

Examples

```
## run EM for marijuana data with ridge prior
data(marijuana)
emResult <- emNorm(marijuana, prior="ridge", prior.df=0.5)

## run MCMC for 5,000 iterations starting from the
## posterior mode using the same prior
mcmcResult <- mcmcNorm(emResult, iter=5000)

## summarize and plot worst linear function
summary(mcmcResult)
plot(mcmcResult$series.worst)
acf(mcmcResult$series.worst, lag.max=50)

## generate 25 multiple imputations, taking
## 100 steps between imputations, and look st
## the first imputed dataset
mcmcResult <- mcmcNorm(emResult, iter=2500, impute.every=100)
mcmcResult$imp.list[[1]]
```

`miInference` *Combine results from analyses after multiple imputation*

Description

This function combines the results from data analyses performed after multiple imputation using methods described by Rubin (1987) and others.

Usage

```
miInference( est.list, std.err.list, method = "scalar",
   df.complete = NULL )

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

Arguments

`est.list`	a list of estimates to be combined. Each component of this list should be a scalar or vector containing point estimates from the analysis of an imputed dataset. This list should have M components, where M is the number of imputations, and all components should have the same length.
`std.err.list`	a list of standard errors to be combined. Each component of this list should be a scalar or vector of standard errors associated with the estimates in `est.list`.

`method`	how are the estimates to be combined? At present, the only type allowed is `"scalar"`, which means that estimands are treated as one-dimensional entities. If `est.list` contains vectors, inference for each element of the vector is carried out separately; covariances among them are not considered.
`df.complete`	degrees of freedom assumed for the complete-data inference. This should be a scalar or a vector of the same length as the components of `est.list` and `std.err.list`.
`x`	a result from `miInference`.
`...`	values to be passed to the methods.

Details

If `df.complete = NULL` or `Inf`, the degrees of freedom are computed by the method of Rubin (1987, Chap.3), which assumes that if there were no missing data, the usual normal approximation for large samples would be appropriate, i.e. that a 95% interval would be computed as the estimate plus or minus 1.96 standard errors. Otherwise, the degrees of freedom are computed by the method of Barnard and Rubin (1999), which assumes that an approximate 95% interval without missing data would be the estimate plus or minus `qt(.975, df.complete)` standard errors.

The result from this function is a list whose `class` attribute has been set to `"miInference"`. If this list is displayed or printed via the generic function `print`, it will be formatted into a table resembling the output from a regression analysis with columns for the estimates, standard errors, t-ratios (estimates divided by their standard errors) and p-values for testing the null hypothesis that each estimate is zero.

Value

a list with the following components:

`names`	character-string labels for the estimands. This is derived from the `names` attribute, if any, of the components of `est.list`.
`est`	combined estimate(s).
`std.err`	standard error(s) for `est`.
`df`	degrees of freedom for Student-t approximation. For example, 95% intervals can be computed as `est` plus or minus `qt(.975,df)*std.err`.
`p`	p-value(s) for testing the null hypothesis that each estimand is zero against a two-tailed alternative.
`rel.incr`	estimated relative increase(s) in variance due to nonresponse.
`mis.inf`	estimated rate(s) of missing information.

Note

Rubin (1987) defined the rate of missing information as `rel.incr + 2/(df+3)` divided by `(rel.incr+1)`, which estimates the information lost due to missing values and due to the fact that the number of multiple imputations is finite. We define it as `rel.incr` divided by `(rel.incr+1)`, the information lost due to missing values, which is consistent with the formulas of Barnard and Rubin (1999).

Author(s)

Joe Schafer <Joseph.L.Schafer@census.gov>

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: Wiley.

For more information about functions in the NORM package, see the manual *NORM Package for R, Version 2* in the subdirectory doc where the library has been installed.

Examples

```
## create 25 multiple imputations for the cholesterol data
## using functions from the NORM library
data(cholesterol)
emResult <- emNorm(cholesterol)
set.seed(234)
mcmcResult <- mcmcNorm(emResult, iter=2500, impute.every=100)

## get estimates and standard errors for the mean of Y1
## minus the mean of Y3
est.list <- as.list(NULL)
std.err.list <- as.list( NULL )
for( m in 1:25 ){
   yimp <- mcmcResult$imp.list[[m]]  # one imputed dataset
   diff <- yimp[,"Y1"] - yimp[,"Y3"]
   est.list[[m]] <- mean(diff)
   std.err.list[[m]] <- sqrt( var(diff) / length(diff) ) }

## combine the results by rules of Barnard and Rubin (1999)
## with df.complete = 27, because a paired t-test in a dataset with
## N=28 cases has 27 degrees of freedom
miResult <- miInference(est.list, std.err.list, df.complete=27)
print(miResult)
```

summary.norm *Summarize information from EM or MCMC algorithms*

Description

Method for summarizing the results from a call to the functions `emNorm` or `mcmcNorm`.

Usage

```
## S3 method for class 'norm'
summary(object, show.variables = (object$method == "EM"),
   show.patterns = (object$method == "EM"),
   show.params = (object$method =="EM"), ...)

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

Arguments

`object`	an object of class `"norm"` which is produced as the result of a call to `emNorm` or `mcmcNorm`; see DETAILS below.
`show.variables`	if `TRUE`, then tables summarizing the response variables and covariates used in the model will be printed.
`show.patterns`	if `TRUE`, then a table summarizing the patterns of missingness will be printed.
`show.params`	if `TRUE`, then the final values of the parameters (estimates after the final iteration of EM or simulated values after the final iteration of MCMC) will be printed.
`x`	a result from `summary.norm`.
`...`	values to be passed to the methods.

Details

The result from a call to `emNorm` or `mcmcNorm` is an object of class `"norm"`, which is a list containing results from the EM or MCMC run. The function `summary.norm`, which is invoked through the generic method `summary`, summarizes the information contained in this object.

The result from `summary.norm` is an object of class `"summary.norm"` which can be displayed or printed via the generic method `print`.

Value

A list that includes all the original components of `obj` plus some additional summaries that are printed via a call to the generic method `print`. These include:

`x.table`	a summary of all variables appearing in the model as predictors or covariates.
`y.table`	a summary of all variables appearing in the model as responses or outcomes.
`em.summary`	a summary of the results from the EM run, including: the number of iterations; whether EM converged; and an empirical estimate of the rate of convergence which estimates the worst fraction of missing information.
`mcmc.summary`	a summary of the results from the MCMC run, including: the number of iterations; whether imputations were created and, if so, how many; and whether parameter series were saved.

Author(s)

Joe Schafer <`Joseph.L.Schafer@census.gov`>

References

For more information about this function and other functions in the norm2 package, see *User's Guide for* norm2 in the library subdirectory doc.

See Also

emNorm, mcmcNorm

Examples

```
## run EM for cholesterol data and summarize
data(cholesterol)
emResult <- emNorm(cholesterol)
summary(emResult)

## run MCMC starting from the ML estimates and summarize
mcmcResult <- mcmcNorm(emResult)
summary(mcmcResult)
```

Index