Package 'mlmi'

June 2, 2023

Type Package

Title Maximum Likelihood Multiple Imputation

Version 1.1.2

Author Jonathan Bartlett

Maintainer Jonathan Bartlett <jonathan.bartlett1@lshtm.ac.uk>

Description Implements so called Maximum Likelihood Multiple Imputation as described by von Hippel and Bartlett (2021) <doi:10.1214/20-STS793>. A number of different imputations are available, by utilising the 'norm', 'cat' and 'mix' packages. Inferences can be performed either using combination rules similar to Rubin's or using a likelihood score based approach based on theory by Wang and Robins (1998) <doi:10.1093/biomet/85.4.935>.

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports MASS, gsl, norm, cat, mix, Matrix, stats, utils, nlme

Suggests bootImpute, testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2023-06-02 12:00:02 UTC

R topics documented:

`catImp` *Imputation for categorical variables using log linear models*

Description

This function performs multiple imputation under a log-linear model as described by Schafer (1997), using his `cat` package, either with or without posterior draws.

Usage

```
catImp(
  obsData,
  M = 10,
  pd = FALSE,
  type = 1,
  margins = NULL,
  steps = 100,
  rseed
)
```

Arguments

`obsData`	The data frame to be imputed. Variables must be coded such that they take consecutive positive integer values, i.e. 1,2,3,...
`M`	Number of imputations to generate.
`pd`	Specify whether to use posterior draws (`TRUE`) or not (`FALSE`).
`type`	An integer specifying what type of log-linear model to impute using. `type=1`, the default, allows for all two-way associations in the log-linear model. `type=2` allows for all three-way associations (plus lower). `type=3` fits a saturated model.
`margins`	An optional argument that can be used instead of `type` to specify the desired log-linear model. See the documentation for the `margins` argument in `ecm.cat` and Schafer (1997) on how to specify this.
`steps`	If `pd` is `TRUE`, the `steps` argument specifies how many MCMC iterations to perform in order to generate the model parameter value for each imputation.
`rseed`	The value to set the `cat` package's random number seed to, using the `rngseed` function of `cat`. This function must be called at least once before imputing using `cat`. If the user wishes to set the seed using `rngseed` before calling `catImp`, set `rseed=NULL`.

Details

By default `catImp` will impute using a log-linear model allowing for all two-way associations, but not higher order associations. This can be modified through use of the `type` and `margins` arguments.

With `pd=FALSE`, all imputed datasets are generated conditional on the MLE of the model parameter, referred to as maximum likelihood multiple imputation by von Hippel and Bartlett (2021).

With `pd=TRUE`, regular 'proper' multiple imputation is used, where each imputation is drawn from a distinct value of the model parameter. Specifically, for each imputation, a single MCMC chain is run, iterating for `steps` iterations.

Imputed datasets can be analysed using `withinBetween`, `scoreBased`, or for example the bootImpute package.

Value

A list of imputed datasets, or if `M=1`, just the imputed data frame.

References

Schafer J.L. (1997). Analysis of incomplete multivariate data. Chapman & Hall, Boca Raton, Florida, USA.

von Hippel P.T. and Bartlett J.W. Maximum likelihood multiple imputation: faster, more efficient imputation without posterior draws. Statistical Science 2021; 36(3) 400-420 doi:10.1214/20STS793.

Examples

```
#simulate a partially observed categorical dataset
set.seed(1234)
n <- 100

#for simplicity we simulate completely independent variables
temp <- data.frame(x1=ceiling(3*runif(n)), x2=ceiling(2*runif(n)), x3=ceiling(2*runif(n)))

#make some data missing
for (i in 1:3) {
  temp[(runif(n)<0.25),i] <- NA
}

#impute using catImp, assuming two-way associations in the log-linear model
imps <- catImp(temp, M=10, pd=FALSE, rseed=4423)

#impute assuming a saturated log-linear model
imps <- catImp(temp, M=10, pd=FALSE, type=3, rseed=4423)
```

ctsTrialWide — *Simulated example data with continuous outcome measured repeatedly over time*

Description

A dataset in the wide form containing simulated data with a repeatedly measured outcome. Some outcome values are missing. The missing data pattern is monotone. There are two baseline covariates.

Usage

```
ctsTrialWide
```

Format

A data frame with 500 rows and 7 variables:

id ID for individual

trt A numeric 0/1 variable indicating control or active treatment group

v A baseline covariate

y0 Baseline measurement of the outcome variable

y1 Outcome measurement at visit 1

y2 Outcome measurement at visit 2

y3 Outcome measurement at visit 3

mixImp — *Imputation for a mixture of continuous and categorical variables using the general location model.*

Description

This function performs multiple imputation under a general location model as described by Schafer (1997), using the `mix` package. Imputation can either be performed using posterior draws (`pd=TRUE`) or conditonal on the maximum likelihood estimate of the model parameters (`pd=FALSE`), referred to as maximum likelihood multiple imputation by von Hippel and Bartlett (2021).

Usage

```
mixImp(
  obsData,
  nCat,
  M = 10,
  pd = FALSE,
  marginsType = 1,
  margins = NULL,
  designType = 1,
  design = NULL,
  steps = 100,
  rseed
)
```

Arguments

obsData	The data frame to be imputed. The categorical variables must be in the first nCat columns, and they must be coded using consecutive positive integers.
nCat	The number of categorical variables in obsData.
M	Number of imputations to generate.
pd	Specify whether to use posterior draws (TRUE) or not (FALSE).
marginsType	An integer specifying what type of log-linear model to use for the categorical variables. marginsType=1, the default, allows for all two-way associations in the log-linear model. marginsType=2 allows for all three-way associations (plus lower). marginsType=3 assumes a saturated log-linear model for the categorical variables.
margins	If marginsType is not specified, margins must be supplied to specify the margins of the log-linear model for the categorical variable. See the help for ecm.mix for details on specifying margins.
designType	An integer specifying how the continuous variables' means should depend on the categorical variables. designType=1, the default, assumes the mean of each continuous variable is a linear function with main effects of the categorical variables. designType=2 assumes each continuous variables has a separate mean for each combination of the categorical variables.
design	If designType is not specified, design must be supplied to specify how the mean of the continuous variables depends on the categorical variables. See the help for ecm.mix for details on specifying design.
steps	If pd is TRUE, the steps argument specifies how many MCMC iterations to perform.
rseed	The value to set the mix package's random number seed to, using the rngseed function of mix. This function must be called at least once before imputing using mix. If the user wishes to set the seed using rngseed before calling mixImp, set rseed=NULL.

Details

See the descriptions for marginsType, margins, designType, design and the documentation in ecm.mix for details about how to specify the model.

Imputed datasets can be analysed using withinBetween, scoreBased, or for example the bootImpute package.

Value

A list of imputed datasets, or if M=1, just the imputed data frame.

References

Schafer J.L. (1997). Analysis of incomplete multivariate data. Chapman & Hall, Boca Raton, Florida, USA.

von Hippel P.T. and Bartlett J.W. Maximum likelihood multiple imputation: faster, more efficient imputation without posterior draws. Statistical Science 2021; 36(3) 400-420 doi:10.1214/20STS793.

Examples

```
#simulate a partially observed dataset with a mixture of categorical and continuous variables
set.seed(1234)

n <- 100

#for simplicity we simulate completely independent categorical variables
x1 <- ceiling(3*runif(n))
x2 <- ceiling(2*runif(n))
x3 <- ceiling(2*runif(n))
y <- 1+0.5*(x1==2)+1.5*(x1==3)+x2+x3+rnorm(n)

temp <- data.frame(x1=x1,x2=x2,x3=x3,y=y)

#make some data missing in all variables
for (i in 1:4) {
  temp[(runif(n)<0.25),i] <- NA
}

#impute conditional on MLE, assuming two-way associations in the log-linear model
#and main effects of categorical variables on continuous one (the default)
imps <- mixImp(temp, nCat=3, M=10, pd=FALSE, rseed=4423)
```

---

normImp *Multivariate normal model imputation*

Description

This function performs multiple imputation under a multivariate normal model as described by Schafer (1997), using his norm package, either with or without posterior draws.

Usage

```
normImp(obsData, M = 10, pd = FALSE, steps = 100, rseed)
```

Arguments

obsData	The data frame to be imputed.
M	Number of imputations to generate.
pd	Specify whether to use posterior draws (TRUE) or not (FALSE).
steps	If pd is TRUE, the steps argument specifies how many MCMC iterations to perform.
rseed	The value to set the norm package's random number seed to, using the rngseed function of norm. This function must be called at least once before imputing using norm. If the user wishes to set the seed using rngseed before calling normImp, set rseed=NULL.

Details

This function imputes from a multivariate normal model with unstructured covariance matrix, as described by Schafer (1997). With pd=FALSE, all imputed datasets are generated conditional on the MLE of the model parameter, referred to as maximum likelihood multiple imputation by von Hippel and Bartlett (2021).

With pd=TRUE, regular 'proper' multiple imputation is used, where each imputation is drawn from a distinct value of the model parameter. Specifically, for each imputation, a single MCMC chain is run, iterating for steps iterations.

Imputed datasets can be analysed using withinBetween, scoreBased, or for example the bootImpute package.

Value

A list of imputed datasets, or if M=1, just the imputed data frame.

References

Schafer J.L. (1997). Analysis of incomplete multivariate data. Chapman & Hall, Boca Raton, Florida, USA.

von Hippel P.T. and Bartlett J.W. Maximum likelihood multiple imputation: faster, more efficient imputation without posterior draws. Statistical Science 2021; 36(3) 400-420 doi:10.1214/20STS793.

Examples

```
#simulate a partially observed dataset from multivariate normal distribution
set.seed(1234)
n <- 100
temp <- MASS::mvrnorm(n=n,mu=rep(0,4),Sigma=diag(4))

#make some values missing
```

```
for (i in 1:4) {
  temp[(runif(n)<0.25),i] <- NA
}

#impute using normImp
imps <- normImp(data.frame(temp), M=10, pd=FALSE, rseed=4423)
```

`normUniImp` *Normal regression imputation of a single variable*

Description

Performs multiple imputation of a single continuous variable using a normal linear regression model. The covariates in the imputation model must be fully observed. By default `normUniImp` imputes every dataset using the maximum likelihood estimates of the imputation model parameters, which here coincides with the OLS estimates, referred to as maximum likelihood multiple imputation by von Hippel and Bartlett (2021). If `pd=TRUE` is specified, it instead performs posterior draw Bayesian imputation.

Usage

```
normUniImp(obsData, impFormula, M = 5, pd = FALSE)
```

Arguments

`obsData`	The data frame to be imputed.
`impFormula`	The linear model formula.
`M`	Number of imputations to generate.
`pd`	Specify whether to use posterior draws (`TRUE`) or not (`FALSE`).

Details

Imputed datasets can be analysed using `withinBetween`, `scoreBased`, or for example the bootImpute package.

Value

A list of imputed datasets, or if `M=1`, just the imputed data frame.

References

von Hippel P.T. and Bartlett J.W. Maximum likelihood multiple imputation: faster, more efficient imputation without posterior draws. Statistical Science 2021; 36(3) 400-420 doi:10.1214/20STS793.

Examples

```
#simulate a dataset with one partially observed (conditionally) normal variable
set.seed(1234)
n <- 100
x <- rnorm(n)
y <- x+rnorm(n)
x[runif(n)<0.25] <- NA
temp <- data.frame(x=x,y=y)

#impute using normImp
imps <- normUniImp(temp, y~x, M=10, pd=FALSE)
```

`refBasedCts` *Reference based imputation of repeated measures continuous data*

Description

Performs multiple imputation of a repeatedly measured continuous endpoint in a randomised clinical trial using reference based imputation as proposed by doi:10.1080/10543406.2013.834911Carpenter et al (2013). This approach can be used for imputation of missing data in randomised clinical trials.

Usage

```
refBasedCts(
  obsData,
  outcomeVarStem,
  nVisits,
  trtVar,
  baselineVars = NULL,
  baselineVisitInt = TRUE,
  type = "MAR",
  M = 5
)
```

Arguments

`obsData`	The data frame to be imputed.
`outcomeVarStem`	String for stem of outcome variable name, e.g. y if y1, y2, y3 are the outcome columns
`nVisits`	The integer number of visits (not including baseline)
`trtVar`	The string variable name of the randomised treatment group variable. The reference arm is assumed to correspond to `trtVar==0`.
`baselineVars`	A string or vector of strings specfying the baseline variables. Often this will include the baseline measurement of the outcome
`baselineVisitInt`	TRUE/FALSE indicating whether to allow for interactions between each baseline variable and visit. Default is TRUE.

type	A string specifying imputation type to use. Valid options are "MAR", "J2R"
M	Number of imputations to generate.

Details

Unlike most implementations of reference based imputation, this implementation imputes conditional on the maximum likelihood estimates of the model parameters, rather than a posterior draw. If one is interested in frequentist valid inferences, this is ok provided the bootstrapping used, for example with using the bootImpute package.

Intermediate missing values are imputed assuming MAR, based on the mixed model fit to that patient's treatment arm. Monotone missing values are imputed using the specified imputation type.

Baseline covariates must be numeric variables. If you have factor variables you must code these into suitable dummy indicators and pass these to the function.

Value

A list of imputed datasets, or if M=1, just the imputed data frame.

References

Carpenter JR, Roger JH, Kenward MG. Analysis of Longitudinal Trials with Protocol Deviation: A Framework for Relevant, Accessible Assumptions, and Inference via Multiple Imputation. (2013) 23(6) 1352-1371

von Hippel PT & Bartlett JW (2019) Maximum likelihood multiple imputation: Faster imputations and consistent standard errors without posterior draws arXiv:1210.0870v10.

Examples

```
#take a look at ctsTrialWide data
head(ctsTrialWide)

#impute the missing outcome values twice assuming MAR
imps <- refBasedCts(ctsTrialWide, outcomeVarStem="y", nVisits=3, trtVar="trt",
                    baselineVars=c("v", "y0"), type="MAR", M=2)

#now impute using jump to reference method
imps <- refBasedCts(ctsTrialWide, outcomeVarStem="y", nVisits=3, trtVar="trt",
                    baselineVars=c("v", "y0"), type="J2R", M=2)

#for frequentist valid inferences we use bootstrapping from the bootImpute package
## Not run:
  #bootstrap 10 times using 2 imputations per bootstrap. Note that to do this
  #we specify nImp=2 to bootImpute by M=1 to the refBasedCts function.
  #Also, 10 bootstraps is far too small to get reliable inferences. To do this
  #for real you would want to use a lot more (e.g. at least nBoot=1000).
  library(bootImpute)
  bootImps <- bootImpute(ctsTrialWide, refBasedCts, nBoot=10, nImp=2,
                         outcomeVarStem="y", nVisits=3, trtVar="trt",
                         baselineVars=c("v", "y0"), type="J2R", M=1)
```

```
  #write a small wrapper function to perform an ANCOVA at the final time point
  ancova <- function(inputData) {
    coef(lm(y3~v+y0+trt, data=inputData))
  }
  ests <- bootImputeAnalyse(bootImps, ancova)
  ests

## End(Not run)
```

---

scoreBased *Score based variance estimation for multiple imputation*

Description

This function implements the score based variance estimation approach described by von Hippel and Bartlett (2021), which is based on earlier work by Wang and Robins (1998).

Usage

```
scoreBased(imps, analysisFun, scoreFun, pd = NULL, dfComplete = NULL, ...)
```

Arguments

imps	A list of imputed datasets produced by one of the imputation functions in mlmi or another package.
analysisFun	A function to analyse the imputed datasets that when applied to a dataset returns a list containing a vector est.
scoreFun	A function whose first argument is a dataset and whose second argument is a vector of parameter values. It should return a matrix of subject level scores evaluated at the parameter value passed to it.
pd	If imps was not generated by one of the imputation functions in mlmi, this argument must be specified to indicate whether the imputations were generated using posterior draws (TRUE) or not (FALSE).
dfComplete	The complete data degrees of freedom. If analysisFun returns a vector of parameter estimates, dfComplete should be a vector of the same length. If not specified, it is assumed that the complete data degrees of freedom is effectively infinite (1e+05).
...	Other parameters that are to be passed through to analysisFun.

Value

A list containing the overall parameter estimates, its corresponding covariance matrix, and degrees of freedom for each parameter.

References

Wang N., Robins J.M. (1998) Large-sample theory for parametric multiple imputation procedures. Biometrika 85(4): 935-948. doi:10.1093/biomet/85.4.935.

von Hippel P.T. and Bartlett J.W. Maximum likelihood multiple imputation: faster, more efficient imputation without posterior draws. Statistical Science 2021; 36(3) 400-420 doi:10.1214/20STS793.

Examples

```
#simulate a partially observed dataset
set.seed(1234)
n <- 100
x <- rnorm(n)
y <- x+rnorm(n)
y[1:50] <- NA
temp <- data.frame(x,y)
#impute using normUniImp, without posterior draws
imps <- normUniImp(temp, y~x, M=10, pd=FALSE)

#define a function which performs our desired analysis on a dataset, returning
#the parameter estimates
yonx <- function(inputData) {
  fitmod <- lm(y~x, data=inputData)
  list(est=c(fitmod$coef,sigma(fitmod)^2))
}

#define a function which when passed a dataset and parameter
#vector, calculates the likelihood score vector
myScore <- function(inputData, parm) {
  beta0 <- parm[1]
  beta1 <- parm[2]
  sigmasq <- parm[3]
  res <- inputData$y - beta0 - beta1*inputData$x
  cbind(res/sigmasq, (res*inputData$x)/sigmasq, res^2/(2*sigmasq^2)-1/(2*sigmasq))
}

#call scoreBased to perform variance estimation
scoreBased(imps, analysisFun=yonx, scoreFun=myScore)
```

`withinBetween` *Within between variance estimation*

Description

This function implements the within-between variance estimation approach. If the imputations were generated using posterior draws, it implements the approach proposed by Barnard & Rubin (1999). If posterior draws were not used, it implements the WB approach described by von Hippel and Bartlett (2021).

Usage

```
withinBetween(imps, analysisFun, pd = NULL, dfComplete = NULL, ...)
```

Arguments

imps	A list of imputed datasets produced by one of the imputation functions in mlmi or another package.
analysisFun	A function to analyse the imputed datasets that when applied to a dataset returns a list containing a vector est and covariance matrix var.
pd	If imps was not generated by one of the imputation functions in mlmi, this argument must be specified to indicate whether the imputations were generated using posterior draws (TRUE) or not (FALSE).
dfComplete	The complete data degrees of freedom. If analysisFun returns a vector of parameter estimates, dfComplete should be a vector of the same length. If not specified, it is assumed that the complete data degrees of freedom is effectively infinite (1e+05).
...	Other parameters that are to be passed through to analysisFun.

Value

A list containing the overall parameter estimates, its corresponding covariance matrix, and degrees of freedom for each parameter.

References

Barnard J, Rubin DB. Miscellanea. Small-sample degrees of freedom with multiple imputation. Biometrika 1999; 86(4): 948-955. doi:10.1093/biomet/86.4.948

von Hippel P.T. and Bartlett J.W. Maximum likelihood multiple imputation: faster, more efficient imputation without posterior draws. Statistical Science 2021; 36(3) 400-420 doi:10.1214/20STS793.

Examples

```
#simulate a partially observed dataset
set.seed(1234)
n <- 100
x <- rnorm(n)
y <- x+rnorm(n)
y[1:50] <- NA
temp <- data.frame(x,y)

#impute using normImp
imps <- normImp(temp, M=100, pd=TRUE, rseed=4423)

#define a function which analyses a dataset using our desired
#analysis model, returning the estimated parameters and their
#corresponding variance covariance matrix
analysisFun <- function(inputData) {
  mod <- lm(y~x, data=inputData)
```

```
  list(est=coef(mod), var=vcov(mod))
}
withinBetween(imps,analysisFun, dfComplete=c(n-2,n-2))
```

Index