Package 'MIBRR'

November 8, 2017

Type Package

Title Multiple Imputation with Bayesian Regularized Regression

Version 0.0.0.9000

Date 2017-11-06

Author Kyle M. Lang [aut, crt]

Maintainer Kyle M. Lang <k.m.lang@uvt.nl>

Description This package implements a multiple imputation method that uses Bayesian regularized regression models as the elementary imputation methods.

License GPL-3 | file LICENSE

Depends mvtnorm, lavaan, mice, optimx

Imports Rcpp (>= 0.11.2), RcppEigen (>= 0.3.2.2.0)

LinkingTo Rcpp, RcppEigen

URL http://github.com/kylelang/MIBRR

BugReports https://github.com/kylelang/MIBRR/issues

R topics documented:

MIBRR-package *Multiple Imputation with Bayesian Regularized Regression*

Description

This package implements a multiple imputation method that uses Bayesian regularized regression models as the elementary imputation methods.

Details

Index: This package was not yet installed at build time.

Author(s)

Kyle M. Lang [aut, crt]

Maintainer: Kyle M. Lang <k.m.lang@uvt.nl>

References

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

Li, Q. and Lin, N. (2010) The Bayesian Elastic Net. *Bayesian Analysis*, **5**(1), 151–170.

Park, T. and Casella, G. (2008) The Bayesian Lasso. *Journal of the American Statistical Association*, **103**, 681–686.

Zhao, Y., and Long, Q. (2013) Multiple imputation in the presence of high-dimensional data. *Statistical Methods in Medical Research*, **0**(0), 1–15.

Examples

```
data(mibrrExampleData)

mibenOut <- miben(data       = mibrrExampleData,
                  nImps      = 100,
                  iterations = c(30, 10),
                  targetVars = c("y", paste0("x", c(1 : 3))),
                  ignoreVars = "idNum")

miblOut <- mibl(data       = mibrrExampleData,
                nImps      = 100,
                iterations = c(50, 10),
                targetVars = c("y", paste0("x", c(1 : 3))),
                ignoreVars = "idNum")

benOut <- ben(data       = mibrrExampleData,
              y          = "y",
              X          = setdiff(colnames(mibrrExampleData), c("y", "idNum")),
```

```
              iterations = c(30, 10)
              )

blOut <- bl(data       = mibrrExampleData,
            y          = "y",
            X          = setdiff(colnames(mibrrExampleData), c("y", "idNum")),
            iterations = c(50, 10)
            )
```

---

ben *Bayesian Elastic Net*

Description

This function will fit the Bayesian elastic net to incomplete data.

Usage

```
ben(data,
    y,
    X              = NULL,
    iterations     = c(100, 10),
    sampleSizes    = list(rep(25, 2), rep(250, 2), rep(500, 2)),
    missCode       = NULL,
    returnConvInfo = TRUE,
    verbose        = TRUE,
    seed           = NULL,
    control        = list()
    )
```

Arguments

data	A, possibly incomplete, numeric data matrix or data frame to which the BEN is to be fit.
y	The column label for the outcome variable.
X	An optional character vector giving the column labels for the predictor variables. When `X = NULL` the target variable is regressed onto all other variables in `data`.
iterations	A two-element numeric vector giving the number of iterations to employ during the MCEM approximation and tuning phases, respectively. Defaults to `iterations = c(100, 10)`.
sampleSizes	A list containing three two-element numeric vectors giving the number of MCMC draws discarded as burn-in and retained, respectively, during the MCEM approximation, tuning, and sampling phases. Defaults to `sampleSizes = list(rep(25, 2), rep(250, 2), rep(500, 2))`.
missCode	An optional integer-valued code used to flag the missing data in `data`. Should take a value that cannot naturally occur in `data`. Not needed when the missing data are coded as `NA`.

`returnConvInfo` A logical switch: Should convergence information for the imputation model (i.e., history of the optimized penalty parameters and R-Hat values for final parameter estimates) be returned?
Defaults to `returnConvInfo = TRUE`.

`verbose` A logical switch: Should verbose output be printed to stdout?
Defaults to `verbose = TRUE`.

`seed` An integer-valued seed for the pseudo-random number generator. When `seed = NULL` R's default PRNG and seed are left alone.

`control` A list of control parameters for the Gibbs sampler and penalty parameter optimization (see Details for more information).

Details

`control` is a list containing the following named elements:

convThresh: The R-Hat value used to judge convergence. R-Hat values < `convThresh` arising during the MCEM tuning phase will trigger a warning.
Defaults to `convThres = 1.1`.

lambda1Starts: An optional numeric vector giving starting values for the LASSO penalty parameter, λ_1. Values are recycled to populate a vector with `size = length(targetVars)`.
Defaults to `rep(0.5, length(targetVars))`.

lambda2Starts: An optional numeric vector giving starting values for the ridge penalty parameter, λ_2. Values are recycled to populate a vector with `size = length(targetVars)`.
Defaults to `rep(0.1 * nPreds, length(targetVars))`, where `nPreds` is the number of predictors in the model.

usePcStarts: A logical switch: Use the starting values for λ_1 suggested by Park and Casella (2008)?
Defaults to `usePcStarts = FALSE`.

smoothingWindow: An integer giving the number of approximation phase Λ values to average over to get the starting $\Lambda's$ for the MCEM tuning phase. Setting `smoothingWindow` > 1 can facilitate convergence of the MCEM tuning phase when burn-in Λ estimates are very noisy.
Defaults to `smoothingWindow = min(10, ceiling(nApprox / 10))` where `nApprox` is the number of MCEM approximation iterations.

center: A logical switch: Should the data be centered before estimating the imputation model? When `center = TRUE` the data centers are added back to the imputed data before the function returns.
Defaults to `center = TRUE`.

scale: A logical switch: Should the predictor data be scaled to have unit variance before estimating the imputation model? When `scale = TRUE` imputed data are reverted to their original scaling before the function returns.
Defaults to `scale = TRUE`.

adaptScales: A logical switch: Should the target variables' scales be actively updated as part of imputation model estimation?
Defaults to `adaptScales = TRUE`.

simpleIntercept: A logical switch: When `simpleIntercept = TRUE`, the mean of each intercept's posterior distribution is taken as $\bar{y}$, otherwise it equals $\bar{y} - \bar{\mathbf{X}}\hat{\beta}$.

minPredCor: The minimum correlation used by `mice::quickpred` when temporarily filling missing data before scaling or when filling missing data on covariates.
Defaults to `minPredCor = 0.3`.

miceIters: The number of iterations used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceIters = 10`.

miceRidge: The ridge penalty used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceRidge = 1e-4`.

miceMethod: The elementary imputation method used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceMethod = "pmm"`.

fimlStarts: A logical switch: Should the model moments from a saturated FIML model be used to scale the target variables? When `fimlStarts = TRUE`, the saturated model is estimated using **lavaan**.
Defaults to `fimlStarts = FALSE`.

optTraceLevel: A non-negative integer passed to the **optimx** `trace` argument. See **optimx** documentation for details.
Defaults to `optTraceLevel = 0`.

optCheckKkt: A logical flag: Should the Kuhn, Jarush, Tucker optimality conditions be checked when optimizing the penalty parameters?
Defaults to `optCheckKkt = TRUE`.

optMethod: A character vector giving the optimization method(s) used by **optimx** to estimate the penalty parameters. Possible options are "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "nlm", "nlminb", "spg", "ucminf", "newuoa", "bobyqa", "nmkb", "hjkb", "Rcgmin", or "Rvmmin". When `length(optMethod) > 1`, **optimx**'s follow-on optimization is employed. See the **optimx** documentation for details.
Defaults to `optMethod = "L-BFGS-B"`.

optBoundLambda: A logical switch: Should the penalty parameters be bounded below by zero?
Defaults to `optBoundLambda = TRUE`.

Value

A list containing the Gibbs samples of all model parameters and whatever additional output is requested via `returnConvInfo`.

Warning

This function is in a highly unstable *alpha* level of development. Please anticipate frequent—and dramatic—changes to the functionality and user interface.

You have been granted access to this package for evaluation purposes, only. This function is **absolutely not** ready for use in real-world analyses!

Author(s)

Kyle M. Lang

References

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

Li, Q. and Lin, N. (2010) The Bayesian Elastic Net. *Bayesian Analysis*, **5**(1), 151–170.

See Also

bl, optimx

Examples

```
data(mibrrExampleData)

benOut <- ben(data       = mibrrExampleData,
              y          = "y",
              X          = setdiff(colnames(mibrrExampleData), c("y", "idNum")),
              iterations = c(30, 10)
              )
```

---

bl *Bayesian LASSO*

Description

This function will fit the Bayesian LASSO to incomplete data.

Usage

```
bl(data,
   y,
   X              = NULL,
   iterations     = c(100, 10),
   sampleSizes    = list(rep(25, 2), rep(250, 2), rep(500, 2)),
   missCode       = NULL,
   returnConvInfo = TRUE,
   verbose        = TRUE,
   seed           = NULL,
   control        = list()
   )
```

Arguments

data	A, possibly incomplete, numeric data matrix or data frame to which the BL is to be fit.
y	The column label for the outcome variable.
X	An optional character vector giving the column labels for the predictor variables. When X = NULL the target variable is regressed onto all other variables in data.

iterations	A two-element numeric vector giving the number of iterations to employ during the MCEM approximation and tuning phases, respectively. Defaults to `iterations = c(100, 10)`.
sampleSizes	A list containing three two-element numeric vectors giving the number of MCMC draws discarded as burn-in and retained, respectively, during the MCEM approximation, tuning, and sampling phases. Defaults to `sampleSizes = list(rep(25, 2), rep(250, 2), rep(500, 2))`.
missCode	An optional integer-valued code used to flag the missing data in `data`. Should take a value that cannot naturally occur in `data`. Not needed when the missing data are coded as `NA`.
returnConvInfo	A logical switch: Should convergence information for the imputation model (i.e., history of the optimized penalty parameters and R-Hat values for final parameter estimates) be returned? Defaults to `returnConvInfo = TRUE`.
verbose	A logical switch: Should verbose output be printed to stdout? Defaults to `verbose = TRUE`.
seed	An integer-valued seed for the pseudo-random number generator. When `seed = NULL` R's default PRNG and seed are left alone.
control	A list of control parameters for the Gibbs sampler and penalty parameter optimization (see Details for more information).

Details

`control` is a list containing the following named elements:

convThresh: The R-Hat value used to judge convergence. R-Hat values < `convThresh` arising during the MCEM tuning phase will trigger a warning.
Defaults to `convThres = 1.1`.

lambda1Starts: An optional numeric vector giving starting values for the LASSO penalty parameter, λ_1. Values are recycled to populate a vector with `size = length(targetVars)`.
Defaults to `rep(0.5, length(targetVars))`.

usePcStarts: A logical switch: Use the starting values for λ_1 suggested by Park and Casella (2008)?
Defaults to `usePcStarts = FALSE`.

smoothingWindow: An integer giving the number of approximation phase Λ values to average over to get the starting $\Lambda's$ for the MCEM tuning phase. Setting `smoothingWindow` > 1 can facilitate convergence of the MCEM tuning phase when burn-in Λ estimates are very noisy.
Defaults to `smoothingWindow = min(10, ceiling(nApprox / 10))` where `nApprox` is the number of MCEM approximation iterations.

center: A logical switch: Should the data be centered before estimating the imputation model? When `center = TRUE` the data centers are added back to the imputed data before the function returns.
Defaults to `center = TRUE`.

scale: A logical switch: Should the predictor data be scaled to have unit variance before estimating the imputation model? When `scale = TRUE` imputed data are reverted to their original scaling before the function returns.
Defaults to `scale = TRUE`.

adaptScales: A logical switch: Should the target variables' scales be actively updated as part of imputation model estimation?
Defaults to `adaptScales = TRUE`.

simpleIntercept: A logical switch: When `simpleIntercept = TRUE`, the mean of each intercept's posterior distribution is taken as $\bar{y}$, otherwise it equals $\bar{y} - \bar{\mathbf{X}}\hat{\beta}$.

minPredCor: The minimum correlation used by `mice::quickpred` when temporarily filling missing data before scaling or when filling missing data on covariates.
Defaults to `minPredCor = 0.3`.

miceIters: The number of iterations used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceIters = 10`.

miceRidge: The ridge penalty used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceRidge = 1e-4`.

miceMethod: The elementary imputation method used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceMethod = "pmm"`.

fimlStarts: A logical switch: Should the model moments from a saturated FIML model be used to scale the target variables? When `fimlStarts = TRUE`, the saturated model is estimated using **lavaan**.
Defaults to `fimlStarts = FALSE`.

Value

A list containing the Gibbs samples of all model parameters and whatever additional output is requested via `returnConvInfo`.

Warning

This function is in a highly unstable *alpha* level of development. Please anticipate frequent—and dramatic—changes to the functionality and user interface.

You have been granted access to this package for evaluation purposes, only. This function is **absolutely not** ready for use in real-world analyses!

Author(s)

Kyle M. Lang

References

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

Park, T. and Casella, G. (2008) The Bayesian Lasso. *Journal of the American Statistical Association*, **103**, 681–686.

Zhao, Y., and Long, Q. (2013) Multiple imputation in the presence of high-dimensional data. *Statistical Methods in Medical Research*, **0**(0), 1–15.

See Also

ben

Examples

```
data(mibrrExampleData)

blOut <- bl(data       = mibrrExampleData,
            y          = "y",
            X          = setdiff(colnames(mibrrExampleData), c("y", "idNum")),
            iterations = c(50, 10)
            )
```

---

miben *Multiple Imputation with the Bayesian Elastic Net*

Description

This function implements MIBEN, a robust MICE-based multiple imputation scheme that employs the Bayesian elastic net as its elementary imputation method.

Usage

```
miben(data,
      nImps,
      targetVars     = NULL,
      ignoreVars     = NULL,
      iterations     = c(100, 10),
      sampleSizes    = list(rep(25, 2), rep(250, 2), rep(500, 2)),
      missCode       = NULL,
      returnConvInfo = TRUE,
      returnParams   = FALSE,
      verbose        = TRUE,
      seed           = NULL,
      control        = list()
      )
```

Arguments

data	An incomplete, numeric data matrix or data frame for which to create the imputations.
nImps	An integer giving the number of imputations to create.
targetVars	An optional character vector giving the column labels for the variables to be imputed. When `targetVars = NULL`, all variables not listed in `ignoreVars` are imputed.
ignoreVars	An optional character vector giving the column labels for those variables that should be excluded from the imputation model (e.g., ID variables).

`iterations` A two-element numeric vector giving the number of iterations to employ during the MCEM approximation and tuning phases, respectively.
Defaults to `iterations = c(100, 10)`.

`sampleSizes` A list containing three two-element numeric vectors giving the number of MCMC draws discarded as burn-in and retained, respectively, during the MCEM approximation, tuning, and sampling phases.
Defaults to `sampleSizes = list(rep(25, 2), rep(250, 2), rep(500, 2))`.

`missCode` An optional integer-valued code used to flag the missing data in `data`. Should take a value that cannot naturally occur in `data`. Not needed when the missing data are coded as `NA`.

`returnConvInfo` A logical switch: Should convergence information for the imputation model (i.e., history of the optimized penalty parameters and R-Hat values for final parameter estimates) be returned?
Defaults to `returnConvInfo = TRUE`.

`returnParams` A logical switch: Should the final Gibbs samples of the imputation model's parameters be returned?
Defaults to `returnParams = FALSE`.

`verbose` A logical switch: Should verbose output be printed to stdout?
Defaults to `verbose = TRUE`.

`seed` An integer-valued seed for the pseudo-random number generator. When `seed = NULL` R's default PRNG and seed are left alone.

`control` A list of control parameters for the Gibbs sampler and penalty parameter optimization (see Details for more information).

Details

`control` is a list containing the following named elements:

convThresh: The R-Hat value used to judge convergence. R-Hat values < `convThresh` arising during the MCEM tuning phase will trigger a warning.
Defaults to `convThres = 1.1`.

lambda1Starts: An optional numeric vector giving starting values for the LASSO penalty parameter, λ_1. Values are recycled to populate a vector with `size = length(targetVars)`.
Defaults to `rep(0.5, length(targetVars))`.

lambda2Starts: An optional numeric vector giving starting values for the ridge penalty parameter, λ_2. Values are recycled to populate a vector with `size = length(targetVars)`.
Defaults to `rep(0.1 * nPreds, length(targetVars))`, where `nPreds` is the number of predictors in the imputation model.

usePcStarts: A logical switch: Use the starting values for λ_1 suggested by Park and Casella (2008)?
Defaults to `usePcStarts = FALSE`.

smoothingWindow: An integer giving the number of approximation phase Λ values to average over to get the starting $\Lambda's$ for the MCEM tuning phase. Setting `smoothingWindow` > 1 can facilitate convergence of the MCEM tuning phase when burn-in Λ estimates are very noisy.
Defaults to `smoothingWindow = min(10, ceiling(nApprox / 10))` where `nApprox` is the number of MCEM approximation iterations.

center: A logical switch: Should the data be centered before estimating the imputation model? When `center = TRUE` the data centers are added back to the imputed data before the function returns.
Defaults to `center = TRUE`.

scale: A logical switch: Should the predictor data be scaled to have unit variance before estimating the imputation model? When `scale = TRUE` imputed data are reverted to their original scaling before the function returns.
Defaults to `scale = TRUE`.

adaptScales: A logical switch: Should the target variables' scales be actively updated as part of imputation model estimation?
Defaults to `adaptScales = TRUE`.

simpleIntercept: A logical switch: When `simpleIntercept = TRUE`, the mean of each intercept's posterior distribution is taken as $\bar{y}$, otherwise it equals $\bar{y} - \bar{\mathbf{X}}\hat{\beta}$.

minPredCor: The minimum correlation used by `mice::quickpred` when temporarily filling missing data before scaling or when filling missing data on covariates.
Defaults to `minPredCor = 0.3`.

miceIters: The number of iterations used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceIters = 10`.

miceRidge: The ridge penalty used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates. Defaults to `miceRidge = 1e-4`.

miceMethod: The elementary imputation method used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceMethod = "pmm"`.

fimlStarts: A logical switch: Should the model moments from a saturated FIML model be used to scale the target variables? When `fimlStarts = TRUE`, the saturated model is estimated using **lavaan**.
Defaults to `fimlStarts = FALSE`.

preserveStructure: A logical switch: Should the data columns be returned in the same order as submitted?
Defaults to `preserveStructure = TRUE`.

optTraceLevel: A non-negative integer passed to the **optimx** `trace` argument. See **optimx** documentation for details.
Defaults to `optTraceLevel = 0`.

optCheckKkt: A logical flag: Should the Kuhn, Jarush, Tucker optimality conditions be checked when optimizing the penalty parameters?
Defaults to `optCheckKkt = TRUE`.

optMethod: A character vector giving the optimization method(s) used by **optimx** to estimate the penalty parameters. Possible options are "Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "nlm", "nlminb", "spg", "ucminf", "newuoa", "bobyqa", "nmkb", "hjkb", "Rcgmin", or "Rvmmin". When `length(optMethod) > 1`, **optimx**'s follow-on optimization is employed. See the **optimx** documentation for details.
Defaults to `optMethod = "L-BFGS-B"`.

optBoundLambda: A logical switch: Should the penalty parameters be bounded below by zero?
Defaults to `optBoundLambda = TRUE`.

Value

A list containing `nImps` imputed versions of `data` and whatever additional output is requested via `returnConvInfo` and `returnParams`.

Warning

This function is in a highly unstable *alpha* level of development. Please anticipate frequent—and dramatic—changes to the functionality and user interface.

You have been granted access to this package for evaluation purposes, only. This function is **absolutely not** ready for use in real-world analyses!

Author(s)

Kyle M. Lang

References

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

Li, Q. and Lin, N. (2010) The Bayesian Elastic Net. *Bayesian Analysis*, **5**(1), 151–170.

See Also

`mibl`, `optimx`

Examples

```
data(mibrrExampleData)

mibenOut <- miben(data       = mibrrExampleData,
                  nImps      = 100,
                  iterations = c(30, 10),
                  targetVars = c("y", paste0("x", c(1 : 3))),
                  ignoreVars = "idNum")
```

`mibl` *Multiple Imputation with the Bayesian LASSO*

Description

This function will implement a MICE-based multiple imputation scheme that employs the Bayesian LASSO as its elementary imputation method.

Usage

```
mibl(data,
     nImps,
     targetVars     = NULL,
     ignoreVars     = NULL,
     iterations     = c(100, 10),
     sampleSizes    = list(rep(25, 2), rep(250, 2), rep(500, 2)),
     missCode       = NULL,
     returnConvInfo = TRUE,
     returnParams   = FALSE,
     verbose        = TRUE,
     seed           = NULL,
     control        = list()
     )
```

Arguments

data	An incomplete, numeric data matrix or data frame for which to create the imputations.
nImps	An integer giving the number of imputations to create.
targetVars	An optional character vector giving the column labels for the variables to be imputed. When `targetVars = NULL`, all variables not listed in `ignoreVars` are imputed.
ignoreVars	An optional character vector giving the column labels for those variables that should be excluded from the imputation model (e.g., ID variables).
iterations	A two-element numeric vector giving the number of iterations to employ during the MCEM approximation and tuning phases, respectively. Defaults to `iterations = c(100, 10)`.
sampleSizes	A list containing three two-element numeric vectors giving the number of MCMC draws discarded as burn-in and retained, respectively, during the MCEM approximation, tuning, and sampling phases. Defaults to `sampleSizes = list(rep(25, 2), rep(250, 2), rep(500, 2))`.
missCode	An optional integer-valued code used to flag the missing data in `data`. Should take a value that cannot naturally occur in `data`. Not needed when the missing data are coded as `NA`.
returnConvInfo	A logical switch: Should convergence information for the imputation model (i.e., history of the optimized penalty parameters and R-Hat values for final parameter estimates) be returned? Defaults to `returnConvInfo = TRUE`.
returnParams	A logical switch: Should the final Gibbs samples of the imputation model's parameters be returned? Defaults to `returnParams = FALSE`.
verbose	A logical switch: Should verbose output be printed to stdout? Defaults to `verbose = TRUE`.
seed	An integer-valued seed for the pseudo-random number generator. When `seed = NULL` R's default PRNG and seed are left alone.

`control` A list of control parameters for the Gibbs sampler and penalty parameter optimization (see Details for more information).

Details

`control` is a list containing the following named elements:

convThresh: The R-Hat value used to judge convergence. R-Hat values < `convThresh` arising during the MCEM tuning phase will trigger a warning.
Defaults to `convThres = 1.1`.

lambda1Starts: An optional numeric vector giving starting values for the LASSO penalty parameter, λ_1. Values are recycled to populate a vector with `size = length(targetVars)`.
Defaults to `rep(0.5, length(targetVars))`.

usePcStarts: A logical switch: Use the starting values for λ_1 suggested by Park and Casella (2008)?
Defaults to `usePcStarts = FALSE`.

smoothingWindow: An integer giving the number of approximation phase Λ values to average over to get the starting $\Lambda's$ for the MCEM tuning phase. Setting `smoothingWindow` > 1 can facilitate convergence of the MCEM tuning phase when burn-in Λ estimates are very noisy.
Defaults to `smoothingWindow = min(10, ceiling(nApprox / 10))` where `nApprox` is the number of MCEM approximation iterations.

center: A logical switch: Should the data be centered before estimating the imputation model? When `center = TRUE` the data centers are added back to the imputed data before the function returns.
Defaults to `center = TRUE`.

scale: A logical switch: Should the predictor data be scaled to have unit variance before estimating the imputation model? When `scale = TRUE` imputed data are reverted to their original scaling before the function returns.
Defaults to `scale = TRUE`.

adaptScales: A logical switch: Should the target variables' scales be actively updated as part of imputation model estimation?
Defaults to `adaptScales = TRUE`.

simpleIntercept: A logical switch: When `simpleIntercept = TRUE`, the mean of each intercept's posterior distribution is taken as $\bar{y}$, otherwise it equals $\bar{y} - \bar{\mathbf{X}}\hat{\beta}$.

minPredCor: The minimum correlation used by `mice::quickpred` when temporarily filling missing data before scaling or when filling missing data on covariates.
Defaults to `minPredCor = 0.3`.

miceIters: The number of iterations used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceIters = 10`.

miceRidge: The ridge penalty used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceRidge = 1e-4`.

miceMethod: The elementary imputation method used by the **mice** package when temporarily filling missing data before scaling or filling missing data on covariates.
Defaults to `miceMethod = "pmm"`.

fimlStarts: A logical switch: Should the model moments from a saturated FIML model be used to scale the target variables? When `fimlStarts = TRUE`, the saturated model is estimated using **lavaan**.
Defaults to `fimlStarts = FALSE`.

preserveStructure: A logical switch: Should the data columns be returned in the same order as submitted?
Defaults to `preserveStructure = TRUE`.

Value

A list containing `nImps` imputed versions of `data` and whatever additional output is requested via `returnConvInfo` and `returnParams`.

Warning

This function is in a highly unstable *alpha* level of development. Please anticipate frequent—and dramatic—changes to the functionality and user interface.

You have been granted access to this package for evaluation purposes, only. This function is **absolutely not** ready for use in real-world analyses!

Author(s)

Kyle M. Lang

References

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

Park, T. and Casella, G. (2008) The Bayesian Lasso. *Journal of the American Statistical Association*, **103**, 681–686.

Zhao, Y., and Long, Q. (2013) Multiple imputation in the presence of high-dimensional data. *Statistical Methods in Medical Research*, **0**(0), 1–15.

See Also

`miben`

Examples

```
data(mibrrExampleData)

miblOut <- mibl(data       = mibrrExampleData,
                nImps      = 100,
                iterations = c(50, 10),
                targetVars = c("y", paste0("x", c(1 : 3))),
                ignoreVars = "idNum")
```

mibrrExampleData *Example Dataset for the MIBRR package.*

Description

Toy data generated as in Experiment 1 of Lang (2015).

Usage

```
data("mibrrExampleData")
```

Format

A data frame with 200 observations on the following 17 variables.

idNum: "Participant" ID Number.

y: Outcome Variable. Contains 20% MAR missingness.

x1–x3: Substantive predictors. Contain 20% MAR missingness.

z1–z12: Exogenous auxiliary variables. Contain 10% MCAR missingness.

Details

These data are only simulated toy data; they have no true meaning. This is one of the datasets generated as part of the Monte Carlo simulation study used to conduct Experiment 1 of my dissertation. The missingness on {y, X} is caused by a linear combination of two randomly selected elements of {Z}, and half of the elements in {Z} have no association with {y, X} (see Lang, 2015, for more details).

Source

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

mibrrL *Print License for* **mibrr**

Description

Print the license under which **mibrr** is distibuted (i.e., the GPL-3).

Usage

```
mibrrL()
```

Value

Print the GPL-3 to stdout.

Author(s)

Kyle M. Lang

Examples

```
mibrrL()
```

---

mibrrW *Print Warranty Statement for* **mibrr**

Description

Print the sections of the GPL-3 that describe the warranty (or complete lack thereof) for **mibrr**.

Usage

```
mibrrW()
```

Value

Text giving the warranty-specific sections of the GPL-3.

Author(s)

Kyle M. Lang

Examples

```
mibrrW()
```

predictMibrr *Generate Posterior Predictions from MIBRR Models*

Description

This function will generate posterior predictive draws form models fit using `miben`, `mibl`, `ben`, or `bl`.

Usage

```
predictMibrr(object,
             newData,
             targetVar = NULL,
             nDraws    = 0)
```

Arguments

`object`	A fitted model object returned by `miben`, `mibl`, `ben`, or `bl`.
`newData`	A data.frame containing new predictor data from which to generate the posterior predictions.
`targetVar`	An optional character vector giving the column labels for the outcome variables for which to generate posterior predictions. When `targetVar = NULL` predictions are generated for all target variables contained in `object`.
`nDraws`	The number of posterior predictive draws to return. Defaults to `nDraws = 0`.

Value

A list containing the posterior predictive draws for each target variable defined in `targetVar` or for all target variables in `object` when `targetVar = NULL`.

Warning

This function is in a highly unstable *alpha* level of development. Please anticipate frequent—and dramatic—changes to the functionality and user interface.

You have been granted access to this package for evaluation purposes, only. This function is **absolutely not** ready for use in real-world analyses!

Note

The column names of `newData` must contain the column names of all variables used to estimate the model represented by `object`.

Author(s)

Kyle M. Lang

References

Lang, K. M. (2015) *MIBEN: Multiple imputation with the Bayesian elastic net* (Unpublished doctoral dissertation). University of Kansas.

See Also

miben, mibl, ben, bl

Examples

```
data(predictData)

## Fit a Bayesian elastic net model:
benOut <- ben(data       = predictData$train,
              y          = "agree",
              X          = setdiff(colnames(predictData$train), "agree"),
              iterations = c(30, 10)
              )

## Generate posterior predictions for 'y':
benPred <- predictMibrr(object = benOut, newData = predictData$test)

## Fit chained Bayesian elastic net models to support MI:
mibenOut <- miben(data         = predictData$incomplete,
                  nImps        = 100,
                  iterations   = c(30, 10),
                  returnParams = TRUE)

## Generate posterior predictions from elementary imputation models:
benPred <- predictMibrr(object = mibenOut, newData = predictData$test)
```

Index