Package 'PcAux'

September 10, 2017

Type Package

Title PcAux: Automatically extract auxiliary features for simple, principled missing data analysis

Version 0.0.0.9007

Date 2017-06-21

Author Kyle M. Lang [aut, crt], Todd D. Little [aut], Steven Chesnut [ctb], Vibhuti Gupta [ctb], Byungkwan Jung [ctb], Pavel Panko [ctb]

Maintainer Kyle M. Lang <kylelang86@gmail.com>

Description

Implements the ideas of Howard, Rhemtulla, and Little (2015) to execute a principled missing data analysis that uses principal component scores as the auxiliary variables. This package extends and early, unpackaged implementation of these ideas that was written by Dr. Steven Chesnut.

License GPL-3 | file LICENSE

Depends mice

Imports methods, vcd, ICC, rlecuyer, parallel

NeedsCompilation no

URL http://github.com/PcAux-Package/PcAux/

BugReports https://github.com/PcAux-Package/PcAux/issues

R topics documented:

PcAux-package *PcAux: Automatically extract auxiliary features for simple, principled missing data analysis*

Description

Implements the ideas of Howard, Rhemtulla, and Little (2015) to execute a principled missing data analysis that uses principal component scores as the auxiliary variables. This package extends and early, unpackaged implementation of these ideas that was written by Dr. Steven Chesnut.

Details

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

Author(s)

Kyle M. Lang [aut, crt], Todd D. Little [aut], Steven Chesnut [ctb], Vibhuti Gupta [ctb], Byungkwan Jung [ctb], Pavel Panko [ctb]

Maintainer: Kyle M. Lang <kylelang86@gmail.com>

References

Howard, W. H., Rhemtulla, M., & Little, T. D. (2015). Using principal components as auxiliary variables in missing data estimation. Multivariate Behavioral Research. 50(3). 285-299.

Examples

```
## Load the data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")

## Create principal component auxiliary variables:
pcAuxOut <- createPcAux(pcAuxData = cleanData,
                        nComps    = c(3, 2)
                        )

## Conduct MI with the pcAux:
miOut <- miWithPcAux(rawData   = iris2,
                     pcAuxData = pcAuxOut,
                     nImps     = 5)
```

```
### OR get the constituent parts ###

## Merge the PC auxiliaries with the original data:
outData <- mergePcAux(pcAuxData = pcAuxOut, rawData = iris2)

### outData can be analyzed via FIML, with the
### pcAux scores used as auxiliary variables.

## Create a predictor matrix:
predMat <- makePredMatrix(mergedData = outData)

### You can run mice() manually by supplying
### predMat to the predictorMatrix argument.
```

`calcTime` *Calculates the time between PcAux processes.*

Description

This function records the system time for the PcAux functions at set intervals to help determine issues with lengthy runs. calcTime allows the user to extract timing information for specific functions of PcAux individually.

Usage

```
calcTime(pcAuxData, what)
```

Arguments

`pcAuxData`	A fitted object of class `PcAuxData` produced as output of `prepData`, `createPcAux`, or `miWithPcAux` functions.
`what`	A character vector indicating the name of a function for which to extract status information. abbreviates the functions `prepData`, `createPcAux`, and `miWithPcAux` as "prep", "create", and "mi", respectively.

Value

A named vector with an entry per interval.

Author(s)

Pavel Panko

Examples

```
## Load the data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")

## Create principal component auxiliary variables:
pcAuxOut <- createPcAux(pcAuxData    = cleanData,
                        nComps       = c(3, 2),
                        interactType = 2)

## Conduct MI with the pcAux:
miOut <- miWithPcAux(rawData   = iris2,
                     pcAuxData = pcAuxOut,
                     nImps     = 5)

## Extract timing infromation:
timeInfo <- calcTime(pcAuxData = miOut, what = "mi")
```

`createPcAux` *Extract Principal Component Auxiliaries for Missing Data Analysis*

Description

Extract principal component scores for use as the auxiliary variables in a principled missing data analysis as described by Howard, Rhemtulla, and Little (2015).

Usage

```
createPcAux(pcAuxData,
            nComps,
            interactType = 1L,
            maxPolyPow   = 3L,
            simMode      = FALSE,
            seed         = NULL,
            verbose      = 2L,
            doImputation = TRUE,
            castData     = !doImputation,
            control,
            ...)
```

Arguments

pcAuxData	An object of class PcAuxData returned from prepData.
nComps	A two-element, numeric vector giving the number of linear and nonlinear, respectively, component scores to extract. See the *Details* section for more information.
interactType	An integer code indicating which method is used to incorporate interactions into the initial, single imputation model. See the *Details* section for more information. Defaults to interactType = 1L.
maxPolyPow	An integer giving the maximum power used when constructing the polynomial terms. Setting maxPolyPow = 1L has the effect of excluding any polynomial terms from the imputation model. Defaults to maxPolyPow = 3L.
simMode	A logical switch turning 'Simulation Mode' on or off. In Simulation Mode all of the automatic data checks will be suppressed. This mode is intended for use when pcAux is being called as part of a Monte Carlo simulation study in which the data properties are well-known by the user. This mode should not be used for 'real-world' data analysis. Defaults to simMode = FALSE.
seed	An optional integer used to seed the random number generator used by the imputation algorithm. Defaults to seed = NULL which leaves the default random number generator unaltered.
verbose	An integer code in 0, 1, 2 defining the verbosity of output printed to the screen. verbose = 0 prints no output; verbose = 1 prints all output except for the messages returned by **mice**; verbose = 2 prints all output, including the messages returned by **mice**. Warnings are always printed, regardless of the value assigned to verbose. Defaults to verbose = 2.
doImputation	A logical switch indicating whether the data should be imputed before extracting the principal component scores. Set to FALSE if the data element in pcAuxData has no missing values (e.g., the imputation was done elsewhere). Defaults to doImputation = TRUE.
castData	A logical switch indicating whether the data element in pcAuxData should have its variables re-typed. Keep as FALSE unless the data have been manipulated after running prepData. Defaults to castData = FALSE.
control	An optional list of control parameters (see 'Details').
...	Not currently used.

Details

The number of component scores requested via the nComps argument can be defined in two ways: as simple counts, or in terms of the proportion of variance in the data explained by the set of extracted components. When specifying nComps, positive integer arguments are interpreted as counts of components; real-valued arguments in [0, 1.0) are interpreted as proportions of variance explained. Additionally, two special options are available. A value of Inf will employ the smallest number of component scores such that adding one more component score does not explain a differentiably larger amount of variance. Specifying any negative value will employ all possible component scores.

The interactType argument can take any of the values in 0, 1, 2, 3. interactType = 0 includes no interaction terms. iteractType = 1 incorporates all two-way interactions between the observed variables and the variables specified in the moderators argument of prepData. iteractType = 2 incorporates all two-way interactions between the linear principal component scores and the variables specified in the moderators argument of prepData. iteractType = 3 incorporates all two-way interactions between linear principal component scores and the raw observed variables. Note that interactType == 2 or interactType == 3 will produce a set of "non-linear" component scores in the pcAux$nonLin field of the PcAuxData object (in addition to the "linear" component scores in the pcAux$lin field), while interactType == 0 and interactType == 1 will produce component scores only in the pcAux$lin field.

The control argument is a key-paired list with the following possible entries:

- miceIters: Number of EM iterations supplied to the maxit argument of mice() during the initial single imputation. Defaults to miceIters = 10L.
- miceRidge: Value of the ridge penalty parameter used to stabilize the imputation models used by mice(). Defaults to miceRidge = 1e-5.
- collinThresh: The strength of linear association used to flag collinear variable for removal. Note that any variable specified in the 'moderators' argument of prepData will be retained, regardless of its collinearity with other variables. Defaults to collinThresh = 0.95.
- minRespCount: The minimum number of observations allowed on each variable without triggering a warning. Defaults to floor(0.05 * nrow(rawData)).
- minPredCor: The minimum magnitude of correlation supplied to the mincor argument of mice::quickpred() when constructing the predictor matrix used by mice() during the initial single imputation. Defaults to minPredCor = 0.1.
- maxNetWts: The maximum number of network weights used by nnet() to fit the polytomous regression models used to impute nominal variables with mice(). Defaults to maxNetWts = 10000L.
- nomMaxLev: The maximum number of response levels for nominal variables that won't trigger a warning. Defaults to nomMaxLev = 10L.
- ordMaxLev: The maximum number of response levels for ordinal variables that won't trigger a warning. Defaults to ordMaxLev = 10L.
- conMinLev: The minimum number of unique responses for continuous variables that won't trigger a warning. Defaults to minConLev = 10L.
- nGVarCats: The number of categories into which continuous grouping variables will be split, if applicable. Defaults to nGVarCats = 3L.
- pcaMemLevel: An integer code representing a trade-off between memory usage and numerical accuracy in the algorithm used to extract the principal component scores. A value of '0L' (the default) will extract the PC scores with the stats::prcomp() package for maximal accuracy. A value of '1L' will use the PcAux:::simplePca() subroutine to extract the PC scores with considerably lower memory usage but, possibly, less numerical accuracy than the prcomp() approach. Leaving this option at the default value should be sufficient for most applications.

Value

An Reference Class object of class PcAuxData with fields for each of the createPcAux function's arguments (except for the raw data which are removed to save resources) and the following modified or additional fields:

- call: A list containing the matched function call to `PcAux`.
- pcAux: A list of length 2. The first element contains the linear principal component auxiliary scores. The second element contains the non-linear principal component auxiliary scores.
- rSquared: A list of length 2. The first element contains the cumulative proportion of variance explained by the linear principal component auxiliary scores. The second element contains the cumulative proportion of variance explained by the non-linear principal component auxiliary scores.
- typeVec: A character vector giving the types assigned to each variable in `rawData`.
- methVec: A character vector giving the elementary imputation methods used by **mice**.
- respCounts: An integer vector giving the variable-wise counts of any missing data in `rawData` that remain after the initial single imputation. Any variables with non-zero entries in respCounts are dropped from the data before extracting the principal component scores to keep the PCA from using listwise-deletion.
- initialPm: A numeric vector giving the initial, variable-wise percents missing for `rawData` before any treatment.
- dropVars: A two-column character matrix. The first column contains the names of all variables dropped from the analysis. The second column contains the reason that the corresponding variable was dropped.
- dummyVars: A character vector containing the names of the dummy-coded representations of the nominal variables.
- probNoms: A character vector giving the variable names for any nominal variables with more levels than `control$nomMaxLev`.
- probOrds: A character vector giving the variable names for any ordinal variables with more levels than `control$ordMaxLev`.
- probCons: A character vector giving the variable names for any continuous variables with fewer levels than `control$conMinLev`.
- levelVec: An integer vector giving the number of unique, non-missing, levels for each variable in `rawData`.
- highPmVars: A character vector containing the names of variables with fewer observed responses than `control$minRespCount`.
- emptyVars: A character vector giving the names of empty columns in `rawData`.
- constants: A character vector giving the names of constant columns in `rawData`.
- collinVars: A three-column character matrix. The first two columns contain the names of pairs of approximately collinear variables. The third column contains their observed linear association.
- impFails: A named list of length 4 with elements: 'firstPass', 'pmm', 'groupMean', and 'grandMean' containing the names of any variables that were not successfully imputed via the named imputation strategy. 'First Pass' imputation refers to the ideal approach that assigns the elementary imputation methods according to each variables declared type. The remaining three methods are less-optimal fall-back approaches.
- patterns: If the imputation process falls back to group mean substitution, this field contains a list of the concatenated grouping patterns used to define the strata within which the group means were computed. This list will have length equal to `length(groupVars)`.

- frozenGVars: If group mean substitution is attempted and some grouping variables are continuous, this field contains the binned versions of the continuous grouping variables that were used for the group mean substitution.
- idFills: A list containing the values used to deterministically fill any missing data that occurred on the ID variables. The length of this argument will equal the number of incomplete ID variables in `rawData`.

Author(s)

Kyle M. Lang

References

Howard, W. H., Rhemtulla, M., & Little, T. D. (2015). Using principal components as auxiliary variables in missing data estimation. Multivariate Behavioral Research. 50(3). 285-299.

See Also

`prepData`, `miWithPcAux`

Examples

```
## Load data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")

## Create the principal component auxiliaries:
pcAuxOut <- createPcAux(pcAuxData = cleanData, nComps = c(3, 0))
```

`getImpData` *Extract multiply imputed datasets from a PcAuxData object.*

Description

This is a simple wrapper function that extracts the completed, multiply imputed data sets from a fitted PcAuxData object produced by running the `miWithPcAux` function.

Usage

```
getImpData(pcAuxData)
```

Arguments

pcAuxData	A fitted object of class PcAuxData produced as output of the miWithPcAux function.

Value

A set of multiply imputed data sets. The format of these data sets is defined by the compFormat value in pcAuxData. See miWithPcAux for more information.

Author(s)

Kyle M. Lang

See Also

miWithPcAux, createPcAux

Examples

```
## Load the data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")

## Create principal component auxiliary variables:
pcAuxOut <- createPcAux(pcAuxData    = cleanData,
                        nComps       = c(3, 2),
                        interactType = 2)

## Conduct MI with the pcAux:
miOut <- miWithPcAux(rawData   = iris2,
                     pcAuxData = pcAuxOut,
                     nImps     = 5)

## Extract a list of imputed data sets:
impList <- getImpData(pcAuxData = miOut)
```

---

inspect *Access fields of a PcAuxData object.*

Description

Provide S3/S4-like access to fields of a PcAuxData Reference Class object.

Usage

```
inspect(object, what)
```

Arguments

object	An initialized RC object of class PcAuxData.
what	A character string naming the field to access in object.

Value

The current value stored in the what field of object.

Author(s)

Kyle M. Lang

Examples

```
## Load data:
data(iris2)

## Prepare the data:
newData <- prepData(rawData   = iris2,
                    nomVars   = "Species",
                    ordVars   = "Petal.Width",
                    idVars    = "ID",
                    dropVars  = "Junk",
                    groupVars = "Species")

## Pull the 'data' field from 'newData':
inspect(object = newData, what = "data")
```

---

iris2 *A modified version of the Fisher/Anderson iris data.*

Description

This is a slight modification of the famous Fisher/Anderson iris data. I've binned petal width and added an ID and junk variable to demonstrate the usage of **package:PcAux** more effectively.

Usage

```
data("iris2")
```

Format

A data frame with 150 observations on the following 7 variables describing the characteristics of a sample of three species of iris.

`ID` A numeric vector of IDs

`Sepal.Length` A numeric vector of sepal lengths

`Sepal.Width` A numeric vector of sepal widths

`Petal.Length` A numeric vector of petal lengths

`Petal.Width` An ordered factor with levels `1` < `2` < `3` < `4` < `5` giving a categorized measure of petal width

`Species` A factor with levels `setosa` `versicolor` `virginica` giving the iris' species

`Junk` A constant nuisance factor with levels `badVar`

Source

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. Annals of Eugenics, 7, 179-188.

The data were collected by: Anderson, Edgar (1935). The irises of the Gaspe Peninsula, Bulletin of the American Iris Society, 59, 2-5.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) The new s language. Wadsworth & Brooks/Cole.

Examples

```
data(iris2)
```

`makePredMatrix` *Make a predictor matrix for use with* **mice**.

Description

Make a predictor matrix for use with **mice** that correctly specifies the auxiliary principal component scores produced by `createPcAux` as the sole predictors in the imputation model.

Usage

```
makePredMatrix(mergedData, nLinear, nNonLinear)
```

Arguments

mergedData	A data frame, such as one returned by PcAux::mergePcAux, containing the incomplete variables to be imputed and the principal component auxiliary variable scores.
nLinear	The number of linear principal component auxiliaries to use as predictors in the imputation model. If not specified, all linear PcAux scores contained in mergedData will be used.
nNonLinear	The number of non-linear principal component auxiliaries to use as predictors in the imputation model. If not specified, all non-linear PcAux scores contained in mergedData will be used.

Value

A pattern matrix with dimensions: c(ncol(mergedData), ncol(mergedData)) that can be supplied to the predictorMatrix argument of **mice**.

Author(s)

Kyle M. Lang

See Also

miWithPcAux

Examples

```
## Load the data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")

## Create principal component auxiliary variables:
pcAuxOut <- createPcAux(pcAuxData = cleanData, nComps = c(3, 0))

## Merge the PC auxiliaries with the original data:
outData <- mergePcAux(pcAuxData = pcAuxOut, rawData = iris2)

## Create a predictor matrix:
predMat <- makePredMatrix(mergedData = outData)
```

mergePcAux *Merge Principal Component Auxiliaries with the raw data from which they were constructed.*

Description

Merge PcAux scores produced by `createPcAux` with a data frame containing the raw data from which the component scores were constructed.

Usage

```
mergePcAux(pcAuxData, rawData, nComps = NULL, verbose = TRUE, ...)
```

Arguments

`pcAuxData`	An object of class *PcAuxData* produced by a call to `createPcAux`.
`rawData`	A data frame containing the raw data used to run `createPcAux`.
`nComps`	A two-element vector giving the number of linear and nonlinear, respectively, component scores to extract. See the *Details* section for more information. When not specified, all component scores that exist in `pcAuxData` are used.
`verbose`	A logical flag indicating whether verbose output should be printed to stdout. Defaults to `verbose = TRUE`.
`...`	Only used when `mergePcAux` is called from within other **PcAux** functions.

Details

This function will attempt to use the ID variables defined in PcAux's `idVars` argument to align rows for merging. If these ID variables are not suitable (i.e., because they don't exist in the raw data or they're not unique row-identifiers), the merging will be accomplished via naive column-binding.

The number of component scores requested via the `nComps` argument can be defined in two ways: as simple counts, or in terms of the proportion of variance in the data explained by the set of extracted components. When specifying `nComps`, integer arguments are interpreted as counts of components; real-valued arguments in [0, 1.0). Additionally, two special options are available. A value of `Inf` will employ the smallest number of component scores such that adding one more component score does not explain a differentiably larger amount of variance. Specifying any negative value will employ all possible component scores. An error is returned when more components are requested than exist in `pcAuxData`.

Value

A data frame with (a subset of) the principal component auxiliary scores from `pcAuxData$pcAux` merged onto the end of the raw data.

Author(s)

Kyle M. Lang

Examples

```
## Load the data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")

## Create principal component auxiliary variables:
pcAuxOut <- createPcAux(pcAuxData    = cleanData,
                        nComps       = c(3, 2),
                        interactType = 3)

## Merge the PC auxiliaries with the original data:
outData <- mergePcAux(pcAuxData = pcAuxOut, rawData = iris2)
```

`miWithPcAux` *Create multiple imputations using the PcAux produced by* `createPcAux`.

Description

Create multiple imputations with the **mice** package using the principal component auxiliary variable scores produced by `createPcAux` as the predictors in the imputation model.

Usage

```
miWithPcAux(rawData,
            pcAuxData,
            nImps      = 100L,
            nomVars    = NULL,
            ordVars    = NULL,
            idVars     = NULL,
            dropVars   = "useExtant",
            nComps     = NULL,
            compFormat = "list",
            seed       = NULL,
            simMode    = FALSE,
            forcePmm   = FALSE,
            nProcess   = 1L,
            verbose    = 2L,
            control)
```

Arguments

rawData	A data frame containing the incomplete data for which to create the multiple imputations.
pcAuxData	An object of class PcAuxData produced by a run of createPcAux.
nImps	An integer giving the number of imputations to create. Defaults to nImps = 100L.
nomVars	An optional character vector containing names of any nominal variables (i.e., unordered factors) that exist in rawData. If unspecified, any nomVars defined in pcAuxData will be used.
ordVars	An optional character vector containing names of any ordinal variables (i.e., ordered factors) that exist in rawData. If unspecified, any ordVars defined in pcAuxData will be used.
idVars	An optional character vector containing names of any ID variables that exist in rawData. Any columns flagged as ID variables should not be represented in nomVars, ordVars, dropVars, or groupVars. If unspecified, any idVars defined in pcAuxData will be used.
dropVars	An optional character vector containing names of any nuisance variables that should be excluded from the imputation process. If unspecified, the default value of dropVars = "useExtant" causes any user-defined dropVars defined in pcAuxData to be used.
nComps	A two-element vector giving the number of linear and nonlinear, respectively, component scores to extract. See the *Details* section for more information. When not specified, all component scores that exist in pcAuxData are used.
compFormat	The format in which the multiply-imputed data sets are returned. Valid arguments are "list", which returns a list of length nImps with each entry containing one imputed data set, "long", "broad", and "repeated". The latter three options are passed directly to the action argument of the mice::complete function. See the documentation for mice::complete for more details on the behavior of the "long", "broad", and "repeated" options. Defaults to compFormat = "list".
seed	An optional integer used to seed the random number generator used by the imputation algorithm. Defaults to seed = NULL which employs any seed defined in createPcAux and, otherwise, leaves the default random number generator unaltered.
simMode	A logical switch turning 'Simulation Mode' on or off. In Simulation Mode all of the automatic data checks will be suppressed. This mode is intended for use when miWithPcAux is being called as part of a Monte Carlo simulation study in which the data properties are well-known by the user. This mode should not be used for 'real-world' data analysis. Defaults to simMode = FALSE.
forcePmm	A logical flag indicating whether or not the imputation should use predictive mean matching as the elementary imputation method for (almost) all variables. If forcePmm == FALSE, the elementary imputation methods are chosen to match each variable's declared type. When forcePmm == TRUE, nominal variables are still imputed with GLM-based methods appropriate for their declared types, but all other variables are imputed with PMM. Defaults to forcePmm = FALSE.

nProcess	An integer that gives the number of parallel processes to use when for parallel MI. Must be less than or equal to the number of available logical processor cores. A value of nProcess = 1L results in serial MI processing. Defaults to nProcess = 1L.
verbose	An integer code in 0, 1, 2 defining the verbosity of output printed to the screen. verbose = 0 prints no output; verbose = 1 prints all output except for the messages returned by **mice**; verbose = 2 prints all output, including the messages returned by **mice**. Warnings are always printed, regardless of the value assigned to verbose. Defaults to verbose = 2.
control	An optional list of control parameters (see 'Details').

Details

The number of component scores requested via the nComps argument can be defined in two ways: as simple counts, or in terms of the proportion of variance in the data explained by the set of extracted components. When specifying nComps, integer arguments are interpreted as counts of components; real-valued arguments in [0, 1.0) are interpreted as proportions of variance explained. Additionally, two special options are available. A value of Inf will employ the smallest number of component scores such that adding one more component score does not explain a differentiably larger amount of variance. Specifying any negative value will employ all possible component scores. An error is returned when more components are requested than exist in pcAuxData.

The control argument is a key-paired list with the following possible entries:

- miceRidge: Value of the ridge penalty parameter used to stabilize the imputation models used by mice(). Defaults to miceRidge = 1e-5.
- minRespCount: The minimum number of observations allowed on each variable without triggering a warning. Defaults to floor(0.05 * nrow(rawData)).
- maxNetWts: The maximum number of network weights used by nnet() to fit the polytomous regression models used to impute nominal variables with mice(). Defaults to maxNetWts = 10000L.
- nomMaxLev: The maximum number of response levels for nominal variables that won't trigger a warning. Defaults to nomMaxLev = 10L.
- ordMaxLev: The maximum number of response levels for ordinal variables that won't trigger a warning. Defaults to ordMaxLev = 10L.
- conMinLev: The minimum number of unique responses for continuous variables that won't trigger a warning. Defaults to minConLev = 10L.

Value

A Reference Class object of class PcAuxData with all of the fields from the object provided to the pcAuxData argument preserved, new fields for each of the miWithPcAux function's arguments and the following modified or additional fields:

- call: A list containing the matched function call to miWithPcAux.
- miDatasets: The completed, multiply imputed data sets. The structure of this field's contents is dictated by the compFormat argument to miWithPcAux.
- miceObject: The mids object returned by **mice** in the process of creating the multiple imputations of rawData.

- nComps: An integer vector of length 2 that contains the number of linear and non-linear, respectively, principal component auxiliary variable scores used as predictors in the multiple imputation models.
- typeVec: A character vector giving the types assigned to each variable in `rawData`.
- methVec: A character vector giving the elementary imputation methods used by **mice**.
- respCounts: An integer vector giving the variable-wise response counts for `rawData`.
- initialPm: A numeric vector giving the initial, variable-wise percents missing for `rawData`, before any treatment.
- dropVars: A two-column character matrix. The first column contains the names of all variables that were excluded from the imputation process (these variables appear in their original, incomplete, form in the multiply imputed data sets). The second column contains the reason that the corresponding variable was excluded.
- probNoms: A character vector giving the variable names for any nominal variables with more levels than `control$nomMaxLev`.
- probOrds: A character vector giving the variable names for any ordinal variables with more levels than `control$ordMaxLev`.
- probCons: A character vector giving the variable names for any continuous variables with fewer levels than `control$conMinLev`.
- levelVec: An integer vector giving the number of unique, non-missing, levels for each column of `rawData`.
- highPmVars: A character vector containing the names of variables with fewer observed responses than `control$minRespCount`.
- emptyVars: A character vector giving the names of empty columns in `rawData`.
- constants: A character vector giving the names of constant columns in `rawData`.

Author(s)

Kyle M. Lang

See Also

`createPcAux`

Examples

```
## Load the data:
data(iris2)

## Prepare the data:
cleanData <- prepData(rawData   = iris2,
                      nomVars   = "Species",
                      ordVars   = "Petal.Width",
                      idVars    = "ID",
                      dropVars  = "Junk",
                      groupVars = "Species")
```

```
## Create principal component auxiliary variables:
pcAuxOut <- createPcAux(pcAuxData    = cleanData,
                        nComps       = c(3, 2),
                        interactType = 2)

## Conduct MI with the pcAux:
miOut <- miWithPcAux(rawData = iris2, pcAuxData = pcAuxOut, nImps = 5)
```

---

pcAuxW *Print warranty statement for* **PcAux**.

Description

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

Usage

```
pcAuxW()
```

Value

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

Author(s)

Kyle M. Lang

Examples

```
## Check PcAux's warranty:
pcAuxW()
```

---

prepData *Prepare Data for Extracting Principal Component Auxiliaries*

Description

Data cleaning to facilitate execution of a principled missing data analysis that uses principal component scores as the auxiliary variables as described by Howard, Rhemtulla, and Little (2015).

Usage

```
prepData(rawData,
         moderators = NULL,
         nomVars    = NULL,
         ordVars    = NULL,
         idVars     = NULL,
         dropVars   = NULL,
         groupVars  = NULL,
         simMode    = FALSE,
         nProcess   = 1L,
         verbose    = 2L,
         control,
         ...)
```

Arguments

rawData	A data frame from which to extract the auxiliary principal components.
moderators	An optional character vector containing names of any moderator variables to include in the initial, single imputation model. The variables supplied here will be interacted with all other observed variables when specifying the initial single imputation model's systematic component. The exact method by which this moderation is incorporated depends on the interactType argument in createPcAux (see the documentation for createPcAux for more information).
nomVars	An optional character vector containing names of any nominal variables (i.e., unordered factors) that exist in rawData.
ordVars	An optional character vector containing names of any ordinal variables (i.e., ordered factors) that exist in rawData.
idVars	An optional character vector containing names of any ID variables that exist in rawData. Any columns flagged as ID variables should not be represented in nomVars, ordVars, dropVars, or groupVars
dropVars	An optional character vector containing names of any nuisance variables that should be dropped before extracting the auxiliary principal component scores.
groupVars	An optional character vector containing names of any grouping variables that can be used to create the strata that define the groups used by the fall-back group-mean substitution. If continuous grouping variables are specified, they are binned via the cut() function with breaks = control$nGVarCuts.
simMode	A logical switch turning 'Simulation Mode' on or off. In Simulation Mode all of the automatic data checks will be suppressed. This mode is intended for use when prepData is being called as part of a Monte Carlo simulation study in which the data properties are well-known by the user. This mode should not be used for 'real-world' data analysis. Defaults to simMode = FALSE.
nProcess	An integer indicating the number of processors to use when using parallel processing for the collinearity checks. A value of nProcess = 1L results in serial processing. Must be less than or equal to the available number of logical processing cores. Defaults to nProcess = 1L.

verbose	An integer code in 0, 1, 2 defining the verbosity of output printed to the screen. `verbose = 0` prints no output; `verbose = 1` prints all output except for the messages returned by **mice**; `verbose = 2` prints all output, including the messages returned by **mice**. Warnings are always printed, regardless of the value assigned to `verbose`. Defaults to `verbose = 2`.
control	An optional list of control parameters (see 'Details').
...	Not currently used.

Details

The `control` argument is a key-paired list with the following possible entries:

- miceIters: Number of EM iterations supplied to the `maxit` argument of mice() during the initial single imputation. Defaults to `miceIters = 10L`.
- miceRidge: Value of the ridge penalty parameter used to stabilize the imputation models used by mice(). Defaults to `miceRidge = 1e-5`.
- collinThresh: The strength of linear association used to flag collinear variable for removal. Defaults to `collinThresh = 0.95`.
- minRespCount: The minimum number of observations allowed on each variable without triggering a warning. Defaults to `floor(0.05 * nrow(rawData))`.
- minPredCor: The minimum magnitude of correlation supplied to the `mincor` argument of mice::quickpred() when constructing the predictor matrix used by mice() during the initial single imputation. Defaults to `minPredCor = 0.1`.
- maxNetWts: The maximum number of network weights used by nnet() to fit the polytomous regression models used to impute nominal variables with mice(). Defaults to `maxNetWts = 10000L`.
- nomMaxLev: The maximum number of response levels for nominal variables that won't trigger a warning. Defaults to `nomMaxLev = 10L`.
- ordMaxLev: The maximum number of response levels for ordinal variables that won't trigger a warning. Defaults to `ordMaxLev = 10L`.
- conMinLev: The minimum number of unique responses for continuous variables that won't trigger a warning. Defaults to `minConLev = 10L`.
- nGVarCats: The number of categories into which continuous grouping variables will be split, if applicable. Defaults to `nGVarCats = 3L`.

Value

An Reference Class object of class PcAuxData with fields for each of the `prepData` function's arguments and the following additional, non-trivial fields:

- call: A list containing the matched function call to `prepData`.
- typeVec: A character vector giving the types assigned to each variable in `rawData`.
- initialPm: A numeric vector giving the initial, variable-wise percents missing for `rawData` before any treatment.
- dropVars: A two-column character matrix. The first column contains the names of all variables dropped from the analysis. The second column contains the reason that the corresponding variable was dropped.

- probNoms: A character vector giving the variable names for any nominal variables with more levels than `control$nomMaxLev`.
- probOrds: A character vector giving the variable names for any ordinal variables with more levels than `control$ordMaxLev`.
- probCons: A character vector giving the variable names for any continuous variables with fewer levels than `control$conMinLev`.
- levelVec: An integer vector giving the number of unique, non-missing, levels for each variable in `rawData`.
- highPmVars: A character vector containing the names of variables with fewer observed responses than `control$minRespCount`.
- emptyVars: A character vector giving the names of empty columns in `rawData`.
- constants: A character vector giving the names of constant columns in `rawData`.
- collinVars: A three-column character matrix. The first two columns contain the names of pairs of approximately collinear variables. The third column contains their observed linear association.
- idFills: A list containing the values used to deterministically fill any missing data that occurred on the ID variables. The length of this argument will equal the number of incomplete ID variables in `rawData`.

Author(s)

Kyle M. Lang

References

Howard, W. H., Rhemtulla, M., & Little, T. D. (2015). Using principal components as auxiliary variables in missing data estimation. Multivariate Behavioral Research. 50(3). 285-299.

See Also

`createPcAux`, `miWithPcAux`

Examples

```
## Load data:
data(iris2)

## Prepare the data:
newData <- prepData(rawData   = iris2,
                    nomVars   = "Species",
                    ordVars   = "Petal.Width",
                    idVars    = "ID",
                    dropVars  = "Junk",
                    groupVars = "Species")
```

Index