Package 'HotDeckImputation'

October 22, 2015

Type Package

Title Hot Deck Imputation Methods for Missing Data

Version 1.1.0

Date 2015-10-21

Depends R (>= 3.0.0), stats

Maintainer Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

Description Hot deck imputation methods to resolve missing data.

License GPL-3

Imports Rglpk

URL https://cran.r-project.org/package=HotDeckImputation,
https://www.researchgate.net/profile/Dieter_Joenssen

NeedsCompilation yes

Author Dieter William Joenssen [aut, cre, cph]

Repository CRAN

Date/Publication 2015-10-22 08:24:04

R topics documented:

HotDeckImputation-package

Hot Deck Imputation Methods for Missing Data

Description

This package provides hot deck imputation methods to resolve missing data. Methods provided are popular in survey methodology, mostly used in the context of large national statistics, but are also finding their way to data mining due to their computational simplicity. A key aspect of this package is the implementation of the commonly advocated donor-limit.

Details

Package:	HotDeckImputation
Type:	Package
Version:	1.1.0
Date:	2014-10-21
License:	GPL-3

`HotDeckImputation` is the ever expanding implementation of hot deck imputation methods, such as the nearest neighbor, the CPS-sequential and random hot deck. The package aims to be comprehensive in the functionality provided, covering key aspects of hot deck imputation not found elsewhere.
Currently implemented functions include:
Nearest neighbor hot deck imputation.
Sequential hot deck imputation.
CPS sequential hot deck imputation.
Development requests are welcome.

Author(s)

Dieter William Joenssen

Maintainer: Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Andridge, R.R. and Little, R.J.A. (2010) A Review of Hot Deck Imputation for Survey Non-response. *International Statistical Review*. **78**, 40–64.

Bankhofer, U. and Joenssen, D.W. (2014) On Limiting Donor Usage for Imputation of Missing Data via Hot Deck Methods. In: M. Spiliopoulou, L. Schmidt-Thieme, and R. Jannings (Eds.): *Data Analysis, Machine Learning and Knowledge Discovery. Studies in Classification, Data Analysis and Knowledge Organization*, 3–11. Berlin/Heidelberg: Springer.

Domschke, W. (1995) *Logistik: Transport*. Munich: Oldenbourg. [in German]

Ford, B. (1983) An Overview of Hot Deck Procedures. In: W. Madow, H. Nisselson and I. Olkin (Eds.): *Incomplete Data in Sample Surveys*. New York: Academic Press, 185–207.

Joenssen, D.W. (2015) Donor Limited Hot Deck Imputation: A Constrained Optimization Problem. In: B. Lausen, S. Krolak-Schwerdt, and M. B\"ohmer (Eds.): *Data Science, Learning by Latent Structures, and Knowledge Discovery. Studies in Classification, Data Analysis and Knowledge Organization*, pages tba. Berlin/Heidelberg: Springer. (in press)

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

Joenssen, D.W. and Bankhofer, U. (2012) Donor Limited Hot Deck Imputation: Effects on Parameter Estimation. *Journal of Theoretical and Applied Computer Science*. **6**, 58–70.

Joenssen, D.W. and Muellerleile, T. (2014) Fehlende Daten bei Data-Mining. *HMD Praxis der Wirtschaftsinformatik*. **51**, 458–468, 2014. doi: 10.1365/s40702-014-0038-8 [in German]

Kalton, G. and Kasprzyk, D. (1986) The Treatment of Missing Survey Data. *Survey Methodology*. **12**, 1–16.

Sande, I. (1983) Hot-Deck Imputation Procedures. In: W. Madow, H. Nisselson and I. Olkin (Eds.): *Incomplete Data in Sample Surveys*. New York: Academic Press, 339–349.

`impute.CPS_SEQ_HD` *CPS Sequential Hot-Deck Imputation*

Description

Resolves missing data by the CPS sequential Hot-Deck Imputation.

Usage

```
impute.CPS_SEQ_HD(DATA = NULL, covariates = NULL, initialvalues = 0,
                  navalues = NA, modifyinplace = TRUE)
```

Arguments

`DATA`	Data containing missing values. Should be a matrix of numbers.
`covariates`	Vector containing the covariates (columns that should be used to create the imputation classes). If `is.null(covariates) | length(covariates)==0` this function defaults to `impute.SEQ_HD`. See Section: Note for further Details.
`initialvalues`	The initial values for the start-up process of the imputation. Should be `"integer"` and `length(initialvalues)==1 | length(initialvalues)==dim(DATA)[2]`. The default of `0` is not normally a good value.
`navalues`	NA code for each variable that should be imputed. Should be `"integer"` and `length(initialvalues)==1 | length(initialvalues)==dim(DATA)[2]`. Default is R's NA value.
`modifyinplace`	Should `DATA` be modified in place? (See the Section: Warning.) If not, a copy is made.

Details

This function imputes the missing values in any variable by creating imputation classes and then replicating the most recently observed value in the class and variable. Imputation classes are created by the adjustment cell method.

Value

An imputed data matrix the same size as the input `DATA`.

Warning

If `modifyinplace == FALSE` `DATA` or rather the variable supplied is edited directly! This is significantly faster if the data set is large.

Note

This is a very fast imputation method. Only one pass of the data is needed. With the use of proper covariates, data may be missing MAR. Covariates should be complete (not missing data). If not, `NA` will be used for building classes. This may or may not be appropriate for the data. The presence of missing values in the covariates in not checked.

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Hanson, R.H. (1978) The Current Population Survey: Design and Methodology. *Technical Paper No. 40* . U.S. Bureau of the Census.

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

Joenssen, D.W. and Bankhofer, U. (2012) Donor Limited Hot Deck Imputation: Effects on Parameter Estimation. *Journal of Theoretical and Applied Computer Science*. **6**, 58–70.

Joenssen, D.W. and Muellerleile, T. (2014) Fehlende Daten bei Data-Mining. *HMD Praxis der Wirtschaftsinformatik*. **51**, 458–468, 2014. doi: 10.1365/s40702-014-0038-8 [in German]

See Also

`impute.SEQ_HD`, `impute.mean`, `impute.NN_HD`

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

n<-1000
m<-3
pmiss<-.1
```

```
#Generate matrix of random integers and 2 binary covariates
Y<-cbind(matrix(sample(0:1,replace=TRUE,size=n*2),nrow=n),
 matrix(sample(0:9,replace=TRUE,size=n*m),nrow=n))

#generate missing values, MCAR, in all but the first two columns
Y[,-c(1,2)][sample(1:length(Y[,-c(1,2)]),
   size=floor(pmiss*length(Y[,-c(1,2)])))]<-NA

#perform the sequential imputation Y within the
#classes created by cross-classifying variables 1 and 2
impute.CPS_SEQ_HD(DATA=Y,covariates=c(1,2),initialvalues=0, navalues=NA, modifyinplace = FALSE)


####an example highlighting the modifyinplace option
#using cbind to show the results of the function and the intial data next to another
cbind(impute.CPS_SEQ_HD(DATA=Y,covariates=c(1,2),initialvalues=0,
                        navalues=NA, modifyinplace = FALSE),Y)
#notice that columns 8-10 (representing Y) still have missing data

#same procedure, except modifyinplace is set to TRUE
cbind(impute.CPS_SEQ_HD(DATA=Y,covariates=c(1,2),initialvalues=0,
                        navalues=NA, modifyinplace = TRUE),Y)
#notice that columns 8-10 (representing Y) are identical to columns 3-5,
#Y has (and any Variables pointing to the same object have) been directly modified.
```

`impute.mean` *Attribute Wise Mean Imputation*

Description

This function imputes the column mean of the complete cases for the missing cases. Utilized by `impute.NN_HD` as a method for dealing with missing values in distance calculation.

Usage

```
impute.mean(DATA = NULL)
```

Arguments

`DATA`	Data with missing values.

Value

Returns an imputed data matrix with the same dimensions as `DATA`.

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Little, R.J.A and Rubin, D.B. (2002) *Statistical Analysis with Missing Data*. Hoboken: Wiley.

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

See Also

impute.NN_HD

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

#Generate matrix of random integers
Y<-matrix(sample(0:9,replace=TRUE,size=6*3),nrow=6)

#generate 6 missing values, MCAR, in all but the first row
Y[-1,][sample(1:12,size=6)]<-NA

#Impute the colMeans of Y
impute.mean(DATA=Y)
```

---

impute.NN_HD *The Nearest Neighbor Hot Deck Algorithms*

Description

A comprehensive function that performs nearest neighbor hot deck imputation. Aspects such as variable weighting, distance types, and donor limiting are implemented. New concepts such as the optimal distribution of donors are also available.

Usage

```
impute.NN_HD(DATA = NULL, distance = "man", weights = "range", attributes = "sim",
 comp = "rw_dist", donor_limit = Inf, optimal_donor = "no",
 list_donors_recipients = NULL, diagnose = NULL)
```

Arguments

DATA	Data containing missing values. Must either be data.frame, then factors and strings will be recoded using model.matrix or Will be coerced by data.matrix.
distance	Distance type to use when searching for the nearest neighbor. See the details section for options.
weights	Weights by which the variables should be scaled. See the details section for options.

`attributes` Determines how attributes should be handled. Currently only "sim", meaning donor and recipient pools are disjoint, is implemented.

`comp` Defines the compensation of missing values for distance calculation. See the details section for options.

`donor_limit` Limits how often a donor may function as such. See the details section for options.

`optimal_donor` Defines how the optimal donor is found when a donor limit is used. See the details section for options.

`list_donors_recipients` Option for manually specifying the donor and recipient pools via a list with components "donors" and "recipients".

`diagnose` Option to recover the generated distances and donor-recipient matches. See details section for usage.

Details

argument: `distance` can be defined as:

- *numeric matrix*, donors x recipients distance matrix
- *numeric length = 1*, Minkovski parameter
- *string length = 1*, distance metric to be used:
  - *"man"*, Manhattan distance
  - *"eukl"*, Euclidean distance
  - *"tscheb"*, Chebyshev distance
  - *"mahal"*, Mahalanobis distance (covariance matrix calculated after missing data compensation, incompatible with comp="rw_dist")

argument: `weights` can be defined as:

- *string length = 1*, reweighting type "range", "sd", "var", or "none"
- *numeric length = 1*, one numeric weight for all variables
- *string vector*, like option 1, only different type for each variable (not implemented)
- *numeric vector*, like option 2, only different numeric weight for each variable
- *list*, mixture of options 3 and 4 (not implemented)

argument: `comp` can be defined as:

- *"rw_dist"*, total distance is reweighted by number of distances that may be computed
- *"mean"*, mean imputation is performed before distance calculation
- *"rseq"*, random hot deck imputation, each variable sequentially (uses impute.SEQ_HD)
- *"rsim"*, random hot deck imputation, each variable simultaneously (not implemented)

argument: `donor_limit` is a single number interpreted by its range:

- *(0,1)*, dynamic donor limit, i.e., .5 means any 1 donor may serve up to 50% of all recipients, rounded up if fractional

- *[1,Inf)*, static donor limit, i.e., 2 means any 1 donor may serve up to 2 recipients, fractional parts discarded
- *Inf*, no donor limit

argument: `optimal_donor` is a single string interpreted by its value:

- *"no"*, donor-recipient matching is performed in order by which the recipients appear in the data (fastest)
- *"rand"*, donor-recipient matching is performed in a random recipient-order
- *"mmin"*, donor-recipient matching is performed by the matrix minimum method (sequence independent)
- *"modifvam"*, donor-recipient matching is performed by a modified (only columns considered) Vogel's approximation method (sequence independent)
- *"vam"*, donor-recipient matching is performed by the Vogel's approximation method (sequence independent)
- *"odd"*, donor-recipient matching is performed via the optimal donor distribution method (sequence independent, best results)

argument: `diagnose` should be:

- `NULL`, no diagnostics will be returned.
- *character string*, desired variable name under which the diagnostics will be saved to `.GlobalEnv`. The following character strings will however default to `NULL` with a warning: "if", "else", "repeat", "while", "function", "for", "in", "next", "break", "TRUE", "FALSE", "NULL", "Inf", "NaN", "NA", "NA_integer_", "NA_real_", "NA_complex_", "NA_character_", "c", "q", "s", "t", "C", "D", "F", "I", "T"
- *anything else*, defaults to `NULL` with a warning.

Should be a character string of the desired variable name which will be created in `.GlobalEnv`

Value

An imputed data matrix the same size as the input `DATA`. If the `diagnose` option is used correctly, a list containing the following components will be created in the workspace:

`distances` the donor-recipient distance matrix used for matching

`list_donors_recipients`
the resultant recipient-donor matches

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Andridge, R.R. and Little, R.J.A. (2010) A Review of Hot Deck Imputation for Survey Non-response. *International Statistical Review*. **78**, 40–64.

Bankhofer, U. and Joenssen, D.W. (2014) On Limiting Donor Usage for Imputation of Missing Data via Hot Deck Methods. In: M. Spiliopoulou, L. Schmidt-Thieme, and R. Jannings (Eds.): *Data Analysis, Machine Learning and Knowledge Discovery. Studies in Classification, Data Analysis and Knowledge Organization*, 3–11. Berlin/Heidelberg: Springer.

Domschke, W. (1995) *Logistik: Transport*. Munich: Oldenbourg. [in German]

Ford, B. (1983) An Overview of Hot Deck Procedures. In: W. Madow, H. Nisselson and I. Olkin (Eds.): *Incomplete Data in Sample Surveys*. New York: Academic Press, 185–207.

Joenssen, D.W. (2015) Donor Limited Hot Deck Imputation: A Constrained Optimization Problem. In: B. Lausen, S. Krolak-Schwerdt, and M. B\"ohmer (Eds.): *Data Science, Learning by Latent Structures, and Knowledge Discovery. Studies in Classification, Data Analysis and Knowledge Organization*, pages 319–328. Berlin/Heidelberg: Springer.

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

Joenssen, D.W. and Bankhofer, U. (2012) Donor Limited Hot Deck Imputation: Effects on Parameter Estimation. *Journal of Theoretical and Applied Computer Science*. **6**, 58–70.

Kalton, G. and Kasprzyk, D. (1986) The Treatment of Missing Survey Data. *Survey Methodology*. **12**, 1–16.

Sande, I. (1983) Hot-Deck Imputation Procedures. In: W. Madow, H. Nisselson and I. Olkin (Eds.): *Incomplete Data in Sample Surveys*. New York: Academic Press, 339–349.

See Also

impute.mean, match.d_r_vam, reweight.data

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

#Generate random integer matrix size 10x4
Y<-matrix(sample(x=1:100,size=10*4),nrow=10)

#remove 5 values, ensuring one complete covariate and 5 donors
Y[-c(1:5),-1][sample(1:15,size=5)]<-NA

#Impute using various different (arbitrarily chosen) settings
impute.NN_HD(DATA=Y,distance="man",weights="var")

impute.NN_HD(DATA=Y,distance=2,weights=rep(.5,4),donor_limit=2,optimal_donor="mmin")

impute.NN_HD(DATA=Y,distance="eukl",weights=.25,comp="mean",donor_limit=1,
 optimal_donor="odd")

#Recover some diagnostics
```

```
impute.NN_HD(DATA=Y,distance="eukl",weights=.25,comp="mean",donor_limit=1,
 optimal_donor="odd",diagnose = "diagnostics")
# look at the diagnostics
 diagnostics
```

`impute.SEQ_HD` *Sequential Hot-Deck Imputation*

Description

Resolves missing data by sequential Hot-Deck Imputation.

Usage

```
impute.SEQ_HD(DATA = NULL, initialvalues = 0, navalues = NA,
              modifyinplace = TRUE)
```

Arguments

`DATA`	Data containing missing values. Should be a matrix of integers.
`initialvalues`	The initial values for the start-up process of the imputation. Should be `"integer"` and `length(initialvalues)==1 | length(initialvalues)==dim(DATA)[2]`. The default of `0` is not normally a good value.
`navalues`	NA code for each variable that should be imputed. Should be `"integer"` and `length(initialvalues)==1 | length(initialvalues)==dim(DATA)[2]`. Default is R's NA value.
`modifyinplace`	Should `DATA` be modified in place? (See the Section: Warning.) If not, a copy is made.

Details

This function imputes the missing values in any variable by replicating the most recently observed value in that variable.

Value

An imputed data matrix the same size as the input `DATA`.

Warning

If `modifyinplace == FALSE` `DATA` or rather the variable supplied is edited directly! This is significantly faster if the data set is large.

Note

This is by far the fastest imputation method. Only one pass of the data is needed. However, no covariate information is used, thus only leads to good results if the data are missing MCAR.

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Hanson, R.H. (1978) The Current Population Survey: Design and Methodology. *Technical Paper No. 40* . U.S. Bureau of the Census.

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

Joenssen, D.W. and Bankhofer, U. (2012) Donor Limited Hot Deck Imputation: Effects on Parameter Estimation. *Journal of Theoretical and Applied Computer Science*. **6**, 58–70.

Joenssen, D.W. and Muellerleile, T. (2014) Fehlende Daten bei Data-Mining. *HMD Praxis der Wirtschaftsinformatik*. **51**, 458–468, 2014. doi: 10.1365/s40702-014-0038-8 [in German]

See Also

impute.CPS_SEQ_HD, impute.mean, impute.NN_HD

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

n<-1000
m<-5
pmiss<-.1


#Generate matrix of random integers
Y<-matrix(sample(0:9,replace=TRUE,size=n*m),nrow=n)

#generate 6 missing values, MCAR, in all but the first row
Y[-1,][sample(1:length(Y[-1,]),size=floor(pmiss*length(Y[-1,])))]<-NA

#perform the sequential imputation Y
impute.SEQ_HD(DATA=Y,initialvalues=0, navalues=NA, modifyinplace = FALSE)

####an example highlighting the modifyinplace option
#using cbind to show the results of the function and the intial data next to another
cbind(impute.SEQ_HD(DATA=Y,initialvalues=0, navalues=NA, modifyinplace = FALSE),Y)
#notice that columns 6-10 (representing Y) still have missing data

#same procedure, except modifyinplace is set to TRUE
cbind(impute.SEQ_HD(DATA=Y,initialvalues=0, navalues=NA, modifyinplace = TRUE),Y)
#notice that columns 6-10 (representing Y) are identical to columns 1-5,
#Y has (and any Variables pointing to the same object have) been directly modified.
```

match.d_r_odd *Donor-Recipient Matching via Optimal Donor Distribution Method*

Description

A function that performs the optimal donor distribution method. GLPK is used to find the optimal solution to the integer program.

Usage

```
match.d_r_odd(distance = NULL, recipients=NULL, donors=NULL, donor_limit=NULL)
```

Arguments

distance	A distance matrix of dimensions length(donors) * length(recipients)
recipients	A vector of object (row) numbers from the original data matrix, indicating which objects require imputation.
donors	A vector of object (row) numbers from the original data matrix, indicating which objects may be used for imputation.
donor_limit	A vector of length(donors) detailing how often any one donor may be used.

Value

A matrix of dimensions length(recipients) * 2. The first column, named recipients, is equivalent to `recipients`. The second column, named donors, contains the donor that is matched to any of the recipients.

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Domschke, W. (1995) *Logistik: Transport*. Munich: Oldenbourg. [in German]

Joenssen, D.W. (2015) Donor Limited Hot Deck Imputation: A Constrained Optimization Problem. In: B. Lausen, S. Krolak-Schwerdt, and M. B\"ohmer (Eds.): *Data Science, Learning by Latent Structures, and Knowledge Discovery. Studies in Classification, Data Analysis and Knowledge Organization*, pages 319–328. Berlin/Heidelberg: Springer.

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

See Also

impute.NN_HD, match.d_r_vam

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

#Set up a random distance matrix
   ndonor=20
   nrecip=20
  distance<-matrix(sample(1:100,replace=TRUE,size=ndonor*nrecip),nrow=ndonor,ncol=nrecip)
#Name donors and recipients
   donors<-1:nrow(distance)
   recipients<-(nrow(distance)+1):(nrow(distance)+ncol(distance))
   colnames(distance)<-recipients
   rownames(distance)<-donors
#Set up the donor limit variable
   donor_limit<-rep(1,length(donors))

#perform the matching
match.d_r_odd(distance=distance,recipients=recipients,donors=donors,
   donor_limit=donor_limit)
```

---

match.d_r_vam *Donor-Recipient Matching via Vogel's Approximation Method*

Description

A function that performs Vogel's approximation method. A heuristic to reduce the total sum of donor-recipient distances.

Usage

```
match.d_r_vam(distance = NULL, recipients=NULL, donors=NULL, donor_limit=NULL)
```

Arguments

distance	A distance matrix of dimensions length(donors) * length(recipients)
recipients	A vector of object (row) numbers from the original data matrix, indicating which objects require imputation.
donors	A vector of object (row) numbers from the original data matrix, indicating which objects may be used for imputation.
donor_limit	A vector of length(donors) detailing how often any one donor may be used.

Value

A matrix of dimensions length(recipients) * 2. The first column, named recipients, is equivalent to `recipients`. The second column, named donors, contains the donor that is matched to any of the recipients.

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

References

Domschke, W. (1995) *Logistik: Transport*. Munich: Oldenbourg. [in German]

Joenssen, D.W. (2015) Donor Limited Hot Deck Imputation: A Constrained Optimization Problem. In: B. Lausen, S. Krolak-Schwerdt, and M. B\"ohmer (Eds.): *Data Science, Learning by Latent Structures, and Knowledge Discovery. Studies in Classification, Data Analysis and Knowledge Organization*, pages 319–328. Berlin/Heidelberg: Springer.

Joenssen, D.W. (2015) Hot-Deck-Verfahren zur Imputation fehlender Daten – Auswirkungen des Donor-Limits. Ilmenau: Ilmedia. [in German, Dissertation]

See Also

impute.NN_HD

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

#Set up a random distance matrix
   ndonor=20
   nrecip=20
  distance<-matrix(sample(1:100,replace=TRUE,size=ndonor*nrecip),nrow=ndonor,ncol=nrecip)
#Name donors and recipients
   donors<-1:nrow(distance)
   recipients<-(nrow(distance)+1):(nrow(distance)+ncol(distance))
   colnames(distance)<-recipients
   rownames(distance)<-donors
#Set up the donor limit variable
   donor_limit<-rep(1,length(donors))

#perform the matching
match.d_r_vam(distance=distance,recipients=recipients,donors=donors,
   donor_limit=donor_limit)
```

---

reweight.data *Reweight Variables*

Description

Reweighting function to preprocess a data matrix prior to Minkovski distance calculation.

Usage

```
reweight.data(DATA = NULL, weights = NULL, minkovski_factor = 1)
```

Arguments

DATA	Data that should be reweighted.
weights	Numeric vector with length equal to the number of variables in DATA.
minkovski_factor	The desired Minkovski parameter that will be used for calculating the distances.

Value

Returns a data matrix with the same dimensions as DATA.

Author(s)

Dieter William Joenssen <Dieter.Joenssen@googlemail.com>

See Also

impute.NN_HD

Examples

```
#Set the random seed to an arbitrary number
set.seed(421)

#Generate matrix of random integers
Y<-matrix(sample(0:9,replace=TRUE,size=6*3),nrow=6)

#choose variable variances
Weights<-1/apply(X=Y,MARGIN=2,FUN=var)

#reweight data for faster Euclidean distance calculation
reweight.data(DATA = Y, weights = Weights, minkovski_factor = 2)
```

Index