Package 'sdcMicro'

May 9, 2026

Type Package

Title Statistical Disclosure Control Methods for Anonymization of Data and Risk Estimation

Version 5.8.1

Date 2026-03-05

Description Data from statistical agencies and other institutions are mostly confidential. This package, introduced in Templ, Kowarik and Meindl (2017) <doi:10.18637/jss.v067.i04>, can be used for the generation of anonymized (micro)data, i.e. for the creation of public- and scientific-use files. The theoretical basis for the methods implemented can be found in Templ (2017) <doi:10.1007/978-3-319-50272-4>. Various risk estimation and anonymization methods are included. Note that the package includes a graphical user interface published in Meindl and Templ (2019) <doi:10.3390/a12090191> that allows to use various methods of this package.

LazyData TRUE

ByteCompile TRUE

LinkingTo Rcpp

Depends R (>= 2.10)

Suggests laeken, parallel, testthat, pdftools, yaml

Imports utils, stats, graphics, car, carData, rmarkdown, knitr, data.table, xtable, robustbase, cluster, MASS, e1071, tools, Rcpp, methods, ggplot2, shiny (>= 1.4.0), haven, rhandsontable, DT, prettydoc, VIM (>= 4.7.0), httr, jsonlite

License GPL-2

URL https://github.com/sdcTools/sdcMicro

Collate '0classes.r' 'addGhostVars.R' 'addNoise.r' 'aux_functions.r' 'createDat.R' 'createNewID.R' 'dataGen.r' 'dataSets.R' 'dRisk.R' 'dRiskRMD.R' 'dUtility.R' 'freqCalc.r' 'globalRecode.R' 'groupAndRename.R' 'GUIfunctions.R'

'indivRisk.R' 'infoLoss.R' 'AI_createSdcObj.R'
'ai_access_utilities.R' 'AI_applyAnonymization.R'
'ai_anonymization_utilities.R' 'LocalRecProg.R' 'localSupp.R'
'localSuppression.R' 'mdav.R' 'measure_risk.R' 'methods.r'
'microaggregation.R' 'modRisk.R'
'muargus_compatibility_functions.R' 'mvTopCoding.R'
'plotFunctions.R' 'plotMicro.R' 'pram.R' 'rankSwap.R'
'RcppExports.R' 'recordSwap.R' 'report.R' 'riskyCells.R'
'sdcMicro-package.R' 'shuffle.R' 'suda2.R' 'timeEstimation.R'
'topBotCoding.R' 'valTable.R' 'zzz.R' 'printFunctions.R'
'mafast.R' 'maG.R' 'sdcApp.R' 'show_sdcMicroObj.R'

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author Matthias Templ [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8638-5276>),
Bernhard Meindl [aut],
Alexander Kowarik [aut] (ORCID:
<https://orcid.org/0000-0001-8598-4130>),
Johannes Gussenbauer [aut],
Organisation For Economic Co-Operation And Development [cph] (Initial published c(++) code (under LGPL) code for rank swapping, mdav-microaggregation, suda2 and other (hierarchical) risk measures),
Statistics Netherlands [cph] (microAggregation cpp code (under EUPL v1.1)),
Pascal Heus [cph] (original measure threshold cpp code (under LGPL))

Maintainer Matthias Templ <matthias.templ@gmail.com>

Repository CRAN

Date/Publication 2026-03-10 08:50:02 UTC

Contents

`addGhostVars` *addGhostVars*

Description

specify variables that are `linked` to a key variable. This results in all suppressions of the key-variable being also applied on the corresponding 'ghost'-variables.

Usage

```
addGhostVars(obj, keyVar, ghostVars)
```

Arguments

`obj`	an object of class `sdcMicroObj-class`
`keyVar`	character-vector of length 1 refering to a categorical key variable within `obj`.
`ghostVars`	a character vector specifying variables that are linked to `keyVar`. Variables listed here must not be be listed in either slots `@keyVars`, `@numVars`, `@pramVars`, `@weightVar`, `@hhId` or `@strataVar` in `obj`.

Value

a modified sdcMicroObj-class object.

Author(s)

Bernhard Meindl

References

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4 doi:10.1007/9783319502724

Examples

```
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
## we want to link the anonymization status of key variabe 'urbrur' to 'hhcivil'
sdc <- addGhostVars(sdc, keyVar="urbrur", ghostVars=c("hhcivil"))
## we want to link the anonymization status of key variabe 'roof' to 'represent'
sdc <- addGhostVars(sdc, keyVar="roof", ghostVars=c("represent"))
```

---

addNoise *Adding noise to perturb data*

Description

Various methods for adding noise to perturb continuous scaled variables.

Usage

```
addNoise(obj, variables = NULL, noise = 150, method = "additive", ...)
```

Arguments

obj	either a data.frame or a sdcMicroObj-class that should be perturbed
variables	vector with names of variables that should be perturbed
noise	amount of noise (in percentages)
method	choose between 'additive', 'correlated', 'correlated2', 'restr', 'ROMM', 'outdect'
...	see possible arguments below

Details

If 'obj' is of class sdcMicroObj-class, all continuous key variables are selected per default. If 'obj' is of class "data.frame" or "matrix", the continuous variables have to be specified.

Method 'additive' adds noise completely at random to each variable depending on its size and standard deviation. 'correlated' and method 'correlated2' adds noise and preserves the covariances as described in R. Brand (2001) or in the reference given below. Method 'restr' takes the sample size into account when adding noise. Method 'ROMM' is an implementation of the algorithm ROMM (Random Orthogonalized Matrix Masking) (Fienberg, 2004). Method 'outdect' adds noise only to outliers. The outliers are identified with univariate and robust multivariate procedures based on a robust mahalanobis distances calculated by the MCD estimator.

Value

If 'obj' was of class sdcMicroObj-class the corresponding slots are filled, like manipNumVars, risk and utility.

If 'obj' was of class "data.frame" or "matrix" an object of class "micro" with following entities is returned:

x	the original data
xm	the modified (perturbed) data
method	method used for perturbation
noise	amount of noise

Author(s)

Matthias Templ and Bernhard Meindl

References

Domingo-Ferrer, J. and Sebe, F. and Castella, J., "On the security of noise addition for privacy in statistical databases", Lecture Notes in Computer Science, vol. 3050, pp. 149-161, 2004. ISSN 0302-9743. Vol. Privacy in Statistical Databases, eds. J. Domingo-Ferrer and V. Torra, Berlin: Springer-Verlag.

Ting, D. Fienberg, S.E. and Trottini, M. "ROMM Methodology for Microdata Release" Joint UN-ECE/Eurostat work session on statistical data confidentiality, Geneva, Switzerland, 2005, https://www.unece.org/fileadmin/DAM/stats/documents/ece/ces/ge.46/2005/wp.11.e.pdf

Ting, D., Fienberg, S.E., Trottini, M. "Random orthogonal matrix masking methodology for microdata release", International Journal of Information and Computer Security, vol. 2, pp. 86-105, 2008.

Templ, M. and Meindl, B., *Robustification of Microdata Masking Methods and the Comparison with Existing Methods*, Lecture Notes in Computer Science, Privacy in Statistical Databases, vol. 5262, pp. 177-189, 2008.

Templ, M. *New Developments in Statistical Disclosure Control and Imputation: Robust Statistics Applied to Official Statistics*, Suedwestdeutscher Verlag fuer Hochschulschriften, 2009, ISBN: 3838108280, 264 pages.

Templ, M. and Meindl, B. and Kowarik, A.: *Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro*, Journal of Statistical Software, 67 (4), 1–36, 2015. doi:10.18637/jss.v067.i04

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4

See Also

sdcMicroObj-class, summary.micro

Examples

```
data(Tarragona)

a1 <- addNoise(Tarragona)
a1


data(testdata)

# donttest because Examples with CPU time > 2.5 times elapsed time
testdata[, c('expend','income','savings')] <-
addNoise(testdata[,c('expend','income','savings')])$xm

## for objects of class sdcMicroObj:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- addNoise(sdc)
```

AI_applyAnonymization *AI_applyAnonymization: Automatically apply anonymization strategy using LLM*

Description

Uses an agentic loop to explore multiple anonymization strategies. The LLM proposes strategies as structured tool calls, each is evaluated with a combined utility score, and the best is selected.

Usage

```
AI_applyAnonymization(
  sdcObj,
  k = 3,
  verbose = TRUE,
  model = NULL,
  api_key = NULL,
```

```
  provider = c("openai", "anthropic", "custom"),
  base_url = NULL,
  confirm = TRUE,
  max_iter = 2,
  n_strategies = 3,
  weights = c(1/3, 1/3, 1/3),
  generateReport = TRUE
)
```

Arguments

sdcObj	An object of class sdcMicroObj.
k	Desired k-anonymity level (default 3).
verbose	If TRUE, prints progress and scores for each strategy.
model	LLM model identifier. If NULL, a default is chosen per provider.
api_key	API key. If NULL, auto-detected from environment variables.
provider	LLM provider: "openai" (default), "anthropic", or "custom" for any OpenAI-compatible endpoint.
base_url	Base URL for the API endpoint. Required when provider = "custom".
confirm	Logical; if TRUE (default) and session is interactive, shows the best strategy and asks for confirmation before applying.
max_iter	Number of refinement iterations after the initial batch (default 2).
n_strategies	Number of strategies in the initial batch (default 3).
weights	Numeric vector of length 3: weights for suppression rate, category loss, and IL1 in the utility score. Default c(1/3, 1/3, 1/3).
generateReport	If TRUE, generates internal and external reports.

Value

Modified sdcMicroObj with the best anonymization strategy applied.

Author(s)

Matthias Templ

Examples

```
## Not run:
if (interactive() && nzchar(Sys.getenv("OPENAI_API_KEY"))) {
  library(sdcMicro)
  data(testdata)
  sdc <- AI_createSdcObj(dat = testdata, policy = "open", confirm = FALSE)
  sdc <- AI_applyAnonymization(sdcObj = sdc, k = 3, verbose = TRUE, confirm = FALSE)
}

## End(Not run)
```

AI_createSdcObj *Create an sdcMicro Object with LLM Assistance*

Description

This function uses a Large Language Model (LLM) to automatically classify variables in a dataset into quasi-identifiers, sensitive variables, numerical variables, and more, and passes the result to `createSdcObj()`. It optionally uses a codebook and policy context.

Usage

```
AI_createSdcObj(
  dat,
  codebook = NULL,
  policy = c("open", "restricted", "confidential"),
  model = NULL,
  api_key = NULL,
  provider = c("openai", "anthropic", "custom"),
  base_url = NULL,
  confirm = TRUE,
  info = TRUE,
  ...
)
```

Arguments

`dat`	A data.frame containing the microdata.
`codebook`	Optional path to a codebook file (currently not parsed; placeholder for future use).
`policy`	Data sharing policy context: `"open"` (default), `"restricted"`, or `"confidential"`.
`model`	The LLM model to use. If `NULL`, a default is chosen per provider.
`api_key`	API key. If `NULL`, auto-detected from environment variables.
`provider`	LLM provider: `"openai"` (default), `"anthropic"`, or `"custom"` for any OpenAI-compatible endpoint (Ollama, Azure, vLLM, Groq, etc.).
`base_url`	Base URL for the API endpoint. Required when `provider = "custom"`.
`confirm`	Logical; if `TRUE` (default) and session is interactive, shows the proposed classification and asks for confirmation before creating the sdcMicroObj.
`info`	Logical; if `TRUE`, prints the LLM classification result and reasoning.
`...`	Additional arguments passed to `createSdcObj()`.

Value

An object of class `sdcMicroObj`.

Author(s)

Matthias Templ

Examples

```
## Not run:
data(testdata)
sdc <- AI_createSdcObj(dat = testdata, policy = "open")
sdc

## End(Not run)
```

`argus_microaggregation` *argus_microaggregation*

Description

calls microaggregation code from mu-argus. In case only one variable should be microaggregated and `useOptimal` is `TRUE`, Hansen-Mukherjee polynomial exact method is applied. In any other case, the Mateo-Domingo method is used.

Usage

```
argus_microaggregation(df, k, useOptimal = FALSE)
```

Arguments

`df`	a `data.frame` with only numerical columns
`k`	required group size
`useOptimal`	(logical) should optimal microaggregation be applied (ony possible in in case of one variable)

Value

a `list` with two elements

- original: the originally provided input data
- microaggregated: the microaggregated data.frame

See Also

mu-Argus manual at https://github.com/sdcTools/manuals/raw/master/mu-argus/MUmanual5.1.pdf

Examples

```
mat <- matrix(sample(1:100, 50, replace=TRUE), nrow=10, ncol=5)
df <- as.data.frame(mat)
res <- argus_microaggregation(df, k=5, useOptimal=FALSE)
```

---

argus_rankswap *argus_rankswap*

Description

argus_rankswap

Usage

```
argus_rankswap(df, perc)
```

Arguments

df	a data.frame with only numerical columns
perc	a number defining the swapping percantage

Value

a list with two elements

- original: the originally provided input data
- swapped: the data.frame containing the swapped values

See Also

mu-Argus manual at https://github.com/sdcTools/manuals/raw/master/mu-argus/MUmanual5.1.pdf

Examples

```
mat <- matrix(sample(1:100, 50, replace=TRUE), nrow=10, ncol=5)
df <- as.data.frame(mat)
res <- argus_rankswap(df, perc=10)
```

`calcRisks` *Recompute Risk and Frequencies for a sdcMicroObj*

Description

Recomputation of Risk should be done after manual changing the content of an object of class sdcMicroObj

Usage

```
calcRisks(obj, ...)
```

Arguments

`obj`	a sdcMicroObj object
`...`	no arguments at the moment

Details

By applying this function, the dislosure risk is re-estimated and the corresponding slots of an object of class sdcMicroObj are updated. This function mostly used internally to automatically update the risk after an sdc method is applied.

Value

a sdcMicroObj object with updated risk values

See Also

sdcMicroObj

Examples

```
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- calcRisks(sdc)
```

casc1 *Small Artificial Data set*

Description

Small Toy Example Data set which was used by Sanz-Mateo et.al.

Format

The format is: int [1:13, 1:7] 10 12 17 21 9 12 12 14 13 15 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:13] "1" "2" "3" "4" ... ..$: chr [1:7] "1" "2" "3" "4" ...

Examples

```
data(casc1)
casc1
```

CASCrefmicrodata *Census data set*

Description

This test data set was obtained on July 27, 2000 using the public use Data Extraction System of the U.S. Bureau of the Census.

Format

A data frame sampled from year 1995 with 1080 observations on the following 13 variables.

AFNLWGT Final weight (2 implied decimal places)

AGI Adjusted gross income

EMCONTRB Employer contribution for hlth insurance

FEDTAX Federal income tax liability

PTOTVAL Total person income

STATETAX State income tax liability

TAXINC Taxable income amount

POTHVAL Total other persons income

INTVAL Amt of interest income

PEARNVAL Total person earnings

FICA Soc. sec. retirement payroll deduction

WSALVAL Amount: Total Wage and salary

ERNVAL Business or Farm net earnings

Source

Public use file from the CASC project. More information on this test data can be found in the paper listed below.

References

Brand, R. and Domingo-Ferrer, J. and Mateo-Sanz, J.M., Reference data sets to test and compare SDC methods for protection of numerical microdata. Unpublished. https://research.cbs.nl/casc/CASCrefmicrodata.pdf

Examples

```
data(CASCrefmicrodata)
str(CASCrefmicrodata)
```

---

createDat *Dummy Dataset for Record Swapping*

Description

createDat() returns dummy data to illustrate targeted record swapping. The generated data contain household ids (‘hid‘), geographic variables (‘nuts1‘, ‘nuts2‘, ‘nuts3‘, ‘lau2‘) as well as some other household or personal variables.

Usage

```
createDat(N = 10000)
```

Arguments

N	integer, number of household to generate

Value

‘data.table‘ containing dummy data

See Also

recordSwap

createNewID — *Creates new randomized IDs*

Description

This is useful if the record IDs consist, for example, of a geo identifier and the household line number. This method can be used to create new, random IDs that cannot be reconstructed.

Usage

```
createNewID(obj, newID, withinVar)
```

Arguments

obj	an sdcMicroObj-class-object
newID	a character specifiying the desired variable name of the new ID
withinVar	if not NULL a character vector specifying a variable (e.g an existing household ID) which will be used when calculating the new IDs. If specified, the same IDs will be assigned to the same values of the given variable.

Value

an sdcMicroObj-class-object with updated slot origData

dataGen — *Fast generation of synthetic data*

Description

Fast generation of (primitive) synthetic multivariate normal data.

Usage

```
dataGen(obj, ...)
```

Arguments

obj	an sdcMicroObj-class-object or a data.frame
...	see possible arguments below

n: amount of observations for the generated data, defaults to 200

use: howto compute covariances in case of missing values, see also argument use in cov. The default choice is 'everything', other possible choices are 'all.obs', 'complete.obs', 'na.or.complete' or 'pairwise.complete.obs'.

Details

Uses the cholesky decomposition to generate synthetic data with approx. the same means and covariances. For details see at the reference.

Value

the generated synthetic data.

Note

With this method only multivariate normal distributed data with approxiomately the same covariance as the original data can be generated without reflecting the distribution of real complex data, which are, in general, not follows a multivariate normal distribution.

Author(s)

Matthias Templ

References

Mateo-Sanz, Martinez-Balleste, Domingo-Ferrer. Fast Generation of Accurate Synthetic Microdata. International Workshop on Privacy in Statistical Databases PSD 2004: Privacy in Statistical Databases, pp 298-306.

See Also

sdcMicroObj-class, shuffle

Examples

```
data(mtcars)

cov(mtcars[,4:6])
cov(dataGen(mtcars[,4:6]))
pairs(mtcars[,4:6])
pairs(dataGen(mtcars[,4:6]))

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- dataGen(sdc)
```

distributeDraws_cpp *Distribute number of swaps*

Description

Distribute number of swaps across lowest hierarchy level according to a predefined `swaprate`. The swaprate is applied such that a single swap counts as swapping 2 households. Number of swaps are randomly rounded up or down, if needed, such that the total number of swaps is in coherence with the swaprate.
NOTE: This is an internal function used for testing the C++-function `distributeDraws` which is used inside the C++-function `recordSwap()`.

Usage

```
distributeDraws_cpp(data, hierarchy, hid, swaprate, seed = 123456L)
```

Arguments

`data`	micro data containing the hierarchy levels and household ID
`hierarchy`	column indices of variables in `data` which refers to the geographic hierarchy in the micro data set. For instance county > municipality > district.
`hid`	column index in `data` which refers to the household identifier.
`swaprate`	double between 0 and 1 defining the proportion of households which should be swapped, see details for more explanations
`seed`	integer setting the sampling seed

distributeRandom_cpp *Distribute*

Description

Distribute ‘totalDraws‘ using ratio/probability vector ‘inputRatio‘ and randomly round each entry up or down such that the distribution results in an integer vector. Returns an integer vector containing the number of units in ‘totalDraws‘ distributetd according to proportions in ‘inputRatio‘.
NOTE: This is an internal function used for testing the C++-function `distributeRandom` which is used inside the C++-function `recordSwap()`.

Usage

```
distributeRandom_cpp(inputRatio, totalDraws, seed)
```

Arguments

inputRatio	vector containing ratios which are used to distribute number units in 'totalDraws'.
totalDraws	number of units to distribute
seed	integer setting the sampling seed

---

dRisk *overal disclosure risk*

Description

Distance-based disclosure risk estimation via standard deviation-based intervals around observations.

Usage

```
dRisk(obj, ...)
```

Arguments

obj	a data.frame or object of class sdcMicroObj-class
...	possible arguments are: xm: perturbed data k: percentage of the standard deviation

Details

An interval (based on the standard deviation) is built around each value of the perturbed value. Then we look if the original values lay in these intervals or not. With parameter k one can enlarge or down scale the interval.

Value

The disclosure risk or/and the modified sdcMicroObj-class

Author(s)

Matthias Templ

References

see method SDID in Mateo-Sanz, Sebe, Domingo-Ferrer. Outlier Protection in Continuous Microdata Masking. International Workshop on Privacy in Statistical Databases. PSD 2004: Privacy in Statistical Databases pp 201-215.

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4

See Also

dUtility

Examples

```
data(free1)
free1 <- as.data.frame(free1)

m1 <- microaggregation(free1[, 31:34], method="onedims", aggr=3)
m2 <- microaggregation(free1[, 31:34], method="pca", aggr=3)
dRisk(obj=free1[, 31:34], xm=m1$mx)
dRisk(obj=free1[, 31:34], xm=m2$mx)
dUtility(obj=free1[, 31:34], xm=m1$mx)
dUtility(obj=free1[, 31:34], xm=m2$mx)

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
## this is already made internally: sdc <- dRisk(sdc)
## and already stored in sdc
```

---

dRiskRMD *RMD based disclosure risk*

Description

Distance-based disclosure risk estimation via robust Mahalanobis Distances.

Usage

```
dRiskRMD(obj, ...)
```

Arguments

obj	an sdcMicroObj-class-object or a data.frame
...	see possible arguments below

xm masked data

k weight for adjusting the influence of the robust Mahalanobis distances, i.e. to increase or decrease each of the disclosure risk intervals.

k2 parameter for method RMDID2 to choose a small interval around each masked observation.

Details

This method is an extension of method SDID because it accounts for the "outlyingness" of each observations. This is a quite natural approach since outliers do have a higher risk of re-identification and therefore these outliers should have larger disclosure risk intervals as observations in the center of the data cloud.

The algorithm works as follows:

1. Robust Mahalanobis distances are estimated in order to get a robust multivariate distance for each observation.

2. Intervals are estimated for each observation around every data point of the original data points where the length of the interval is defined/weighted by the squared robust Mahalanobis distance and the parameter k. The higher the RMD of an observation the larger the interval.

3. Check if the corresponding masked values fall into the intervals around the original values or not. If the value of the corresponding observation is within such an interval the whole observation is considered unsafe. So, we get a whole vector indicating which observation is save or not, and we are finished already when using method RMDID1).

4. For method RMDID1w: we return the weighted (via RMD) vector of disclosure risk.

5. For method RMDID2: whenever an observation is considered unsafe it is checked if m other observations from the masked data are very close (defined by a parameter $k2$ for the length of the intervals as for SDID or RSDID) to such an unsafe observation from the masked data, using Euclidean distances. If more than m points are in such a small interval, we conclude that this observation is "save".

Value

The disclosure risk or the modified `sdcMicroObj-class`

`risk1`	percentage of sensitive observations according to method RMDID1.
`risk2`	standardized version of risk1
`wrisk1`	amount of sensitive observations according to RMDID1 weighted by their corresponding robust Mahalanobis distances.
`wrisk2`	RMDID2 measure
`indexRisk1`	index of observations with high risk according to risk1 measure
`indexRisk2`	index of observations with high risk according to wrisk2 measure

Author(s)

Matthias Templ

References

Templ, M. and Meindl, B., *Robust Statistics Meets SDC: New Disclosure Risk Measures for Continuous Microdata Masking*, Lecture Notes in Computer Science, Privacy in Statistical Databases, vol. 5262, pp. 113-126, 2008.

Templ, M. *New Developments in Statistical Disclosure Control and Imputation: Robust Statistics Applied to Official Statistics*, Suedwestdeutscher Verlag fuer Hochschulschriften, 2009, ISBN: 3838108280, 264 pages.

See Also

dRisk

Examples

```
data(Tarragona)
x <- Tarragona[, 5:7]
y <- addNoise(x)$xm
dRiskRMD(x, xm=y)
dRisk(x, xm=y)

data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')

sdc <- dRiskRMD(sdc)
```

---

dUtility *Data-Utility measures*

Description

dUtility() allows to compute different measures of data-utility based on various distances using original and perturbed variables.

Usage

```
dUtility(obj, ...)
```

Arguments

obj	original data or object of class sdcMicroObj
...	see arguments below • xm: perturbed data • method: method IL1, IL1s or eigen. More methods are implemented in summary.micro()

Details

The standardised distances of the perturbed data values to the original ones are measured. The following measures are available:

- "IL1: sum of absolute distances between original and perturbed variables scaled by absolute values of the original variables
- "IL1s: measures the absolute distances between original and perturbed ones, scaled by the standard deviation of original variables times the square root of 2.
- "eigen; compares the eigenvalues of original and perturbed data
- "robeigen; compares robust eigenvalues of original and perturbed data

Value

data utility or modified entry for data utility the sdcMicroObj.

Author(s)

Matthias Templ

References

for IL1 and IL1s: see Mateo-Sanz, Sebe, Domingo-Ferrer. Outlier Protection in Continuous Microdata Masking. International Workshop on Privacy in Statistical Databases. PSD 2004: Privacy in Statistical Databases pp 201-215.

Templ, M. and Meindl, B., Robust Statistics Meets SDC: New Disclosure Risk Measures for Continuous Microdat
Lecture Notes in Computer Science, Privacy in Statistical Databases, vol. 5262, pp. 113-126, 2008.

See Also

dRisk(), dRiskRMD()

Examples

```
data(free1)
free1 <- as.data.frame(free1)

m1 <- microaggregation(free1[, 31:34], method="onedims", aggr=3)
m2 <- microaggregation(free1[, 31:34], method="pca", aggr=3)
dRisk(obj=free1[, 31:34], xm=m1$mx)
dRisk(obj=free1[, 31:34], xm=m2$mx)
dUtility(obj=free1[, 31:34], xm=m1$mx)
dUtility(obj=free1[, 31:34], xm=m2$mx)
data(Tarragona)
x <- Tarragona[, 5:7]
y <- addNoise(x)$xm
dRiskRMD(x, xm=y)
dRisk(x, xm=y)
dUtility(x, xm = y, method = "IL1")
dUtility(x, xm = y, method = "IL1s")
dUtility(x, xm = y, method = "eigen")
dUtility(x, xm = y, method = "robeigen")

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
## this is already made internally, so you don't need to run this:
sdc <- dUtility(sdc)
```

EIA *EIA data set*

Description

Data set obtained from the U.S. Energy Information Authority.

Format

A data frame with 4092 observations on the following 15 variables.

UTILITYID UNIQUE UTILITY IDENTIFICATION NUMBER

UTILNAME UTILITY NAME. A factor with levels 4-County Electric Power Assn Alabama Power Co Alaska Electric Appalachian Electric Coop Appalachian Power Co Arizona Public Service Co Arkansas Power & Light Co Arkansas Valley Elec Coop Corp Atlantic City Electric Company Baker Electric Coop Inc Baltimore Gas & Electric Co Bangor Hydro-Electric Co Berkeley Electric Coop Inc Black Hills Corp Blackstone Valley Electric Co Bonneville Power Admin Boston Edison Co Bountiful City Light & Power Bristol City of Brookings City of Brunswick Electric Member Corp Burlington City of Carolina Power & Light Co Carroll Electric Coop Corp Cass County Electric Coop Inc Central Illinois Light Company Central Illinois Pub Serv Co Central Louisiana Elec Co Inc Central Maine Power Co Central Power & Light Co Central Vermont Pub Serv Corp Chattanooga City of Cheyenne Light Fuel & Power Co Chugach Electric Assn Inc Cincinnati Gas & Electric Co Citizens Utilities Company City of Boulder City City of Clinton City of Dover City of Eugene City of Gillette City of Groton Dept of Utils City of Idaho Falls City of Independence City of Newark City of Reading City of Tupelo Water & Light D Clarksville City of Cleveland City of Cleveland Electric Illum Co Coast Electric Power Assn Cobb Electric Membership Corp Colorado River Commission Colorado Springs City of Columbus Southern Power Co Commonwealth Edison Co Commonwealth Electric Co Connecticut Light & Power Co Consolidated Edison Co-NY Inc Consumers Power Co Cornhusker Public Power Dist Cuivre River Electric Coop Inc Cumberland Elec Member Corp Dakota Electric Assn Dawson County Public Pwr Dist Dayton Power & Light Company Decatur City of Delaware Electric Coop Inc Delmarva Power & Light Co Detroit Edison Co Duck River Elec Member Corp Duke Power Co Duquesne Light Company East Central Electric Assn Eastern Maine Electric Coop El Paso Electric Co Electric Energy Inc Empire District Electric Co Exeter & Hampton Electric Co Fairbanks City of Fayetteville Public Works Comm First Electric Coop Corp Florence City of Florida Power & Light Co Florida Power Corp Fort Collins Lgt & Pwr Utility Fremont City of Georgia Power Co Gibson County Elec Member Corp Golden Valley Elec Assn Inc Grand Island City of Granite State Electric Co Green Mountain Power Corp Green River Electric Corp Greeneville City of Gulf Power Company Gulf States Utilities Co Hasting Utilities Hawaii Electric Light Co Inc Hawaiian Electric Co Inc Henderson-Union Rural E C C Homer Electric Assn Inc Hot Springs Rural El Assn Inc Houston Lighting & Power Co Huntsville City of Idaho Power Co IES Utilities Inc Illinois Power Co Indiana Michigan Power Co Indianapolis Power & Light Co Intermountain Rural Elec Assn Interstate Power Co Jackson Electric Member Corp Jersey Central Power&Light Co Joe Wheeler Elec Member Corp Johnson City City of Jones-Onslow Elec Member Corp Kansas City City of Kansas City Power & Light

Co Kentucky Power Co Kentucky Utilities Co Ketchikan Public Utilities Kingsport Power Co Knoxville City of Kodiak Electric Assn Inc Kootenai Electric Coop, Inc Lansing Board of Water & Light Lenoir City City of Lincoln City of Long Island Lighting Co Los Angeles City of Louisiana Power & Light Co Louisville Gas & Electric Co Loup River Public Power Dist Lower Valley Power & Light Inc Maine Public Service Company Massachusetts Electric Co Matanuska Electric Assn Inc Maui Electric Co Ltd McKenzie Electric Coop Inc Memphis City of MidAmerican Energy Company Middle Tennessee E M C Midwest Energy, Inc Minnesota Power & Light Co Mississippi Power & Light Co Mississippi Power Co Monongahela Power Co Montana-Dakota Utilities Co Montana Power Co Moon Lake Electric Assn Inc Narragansett Electric Co Nashville City of Nebraska Public Power District Nevada Power Co New Hampshire Elec Coop, Inc New Orleans Public Service Inc New York State Gas & Electric Newport Electric Corp Niagara Mohawk Power Corp Nodak Rural Electric Coop Inc Norris Public Power District Northeast Oklahoma Electric Co Northern Indiana Pub Serv Co Northern States Power Co Northwestern Public Service Co Ohio Edison Co Ohio Power Co Ohio Valley Electric Corp Oklahoma Electric Coop, Inc Oklahoma Gas & Electric Co Oliver-Mercer Elec Coop, Inc Omaha Public Power District Otter Tail Power Co Pacific Gas & Electric Co Pacificorp dba Pacific Pwr & L Palmetto Electric Coop, Inc Pennsylvania Power & Light Co Pennyrile Rural Electric Coop Philadelphia Electric Co Pierre Municipal Electric Portland General Electric Co Potomac Edison Co Potomac Electric Power Co Poudre Valley R E A, Inc Power Authority of State of NY Provo City Corporation Public Service Co of Colorado Public Service Co of IN Inc Public Service Co of NH Public Service Co of NM Public Service Co of Oklahoma Public Service Electric&Gas Co PUD No 1 of Clark County PUD No 1 of Snohomish County Puget Sound Power & Light Co Rappahannock Electric Coop Rochester Public Utilities Rockland Electric Company Rosebud Electric Coop Inc Rutherford Elec Member Corp Sacramento Municipal Util Dist Salmon River Electric Coop Inc Salt River Proj Ag I & P Dist San Antonio City of Savannah Electric & Power Co Seattle City of Sierra Pacific Power Co Singing River Elec Power Assn Sioux Valley Empire E A Inc South Carolina Electric&Gas Co South Carolina Pub Serv Auth South Kentucky Rural E C C Southern California Edison Co Southern Nebraska Rural P P D Southern Pine Elec Power Assn Southwest Tennessee E M C Southwestern Electric Power Co Southwestern Public Service Co Springfield City of St Joseph Light & Power Co State Level Adjustment Tacoma City of Tampa Electric Co Texas-New Mexico Power Co Texas Utilities Electric Co Tri-County Electric Assn Inc Tucson Electric Power Co Turner-Hutchinsin El Coop, Inc TVA U S Bureau of Indian Affairs Union Electric Co Union Light Heat & Power Co United Illuminating Co Upper Cumberland E M C UtiliCorp United Inc Verdigris Valley Electric Coop Verendrye Electric Coop Inc Virginia Electric & Power Co Volunteer Electric Coop Wallingford Town of Warren Rural Elec Coop Corp Washington Water Power Co Watertown Municipal Utils Dept Wells Rural Electric Co West Penn Power Co West Plains Electric Coop Inc West River Electric Assn, Inc Western Massachusetts Elec Co Western Resources Inc Wheeling Power Company Wisconsin Electric Power Co Wisconsin Power & Light Co Wisconsin Public Service Corp Wright-Hennepin Coop Elec Assn Yellowstone Vlly Elec Coop Inc

STATE STATE FOR WHICH THE UTILITY IS REPORTING. A factor with levels AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC ND NE NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA VT WA WI WV WY

YEAR REPORTING YEAR FOR THE DATA

MONTH REPORTING MONTH FOR THE DATA

RESREVENUE REVENUE FROM SALES TO RESIDENTIAL CONSUMERS

RESSALES SALES TO RESIDENTIAL CONSUMERS

COMREVENUE REVENUE FROM SALES TO COMMERCIAL CONSUMERS

COMSALES SALES TO COMMERCIAL CONSUMERS

INDREVENUE REVENUE FROM SALES TO INDUSTRIAL CONSUMERS

INDSALES SALES TO INDUSTRIAL CONSUMERS

OTHREVENUE REVENUE FROM SALES TO OTHER CONSUMERS

OTHRSALES SALES TO OTHER CONSUMERS

TOTREVENUE REVENUE FROM SALES TO ALL CONSUMERS

TOTSALES SALES TO ALL CONSUMERS

Source

Public use file from the CASC project.

References

Brand, R. and Domingo-Ferrer, J. and Mateo-Sanz, J.M., Reference data sets to test and compare SDC methods for protection of numerical microdata. Unpublished. https://research.cbs.nl/casc/CASCrefmicrodata.pdf

Examples

```
data(EIA)
head(EIA)
```

---

extractManipData *Remove certain variables from the data set inside a sdc object.*

Description

Extract the manipulated data from an object of class sdcMicroObj-class

Usage

```
extractManipData(
  obj,
  ignoreKeyVars = FALSE,
  ignorePramVars = FALSE,
  ignoreNumVars = FALSE,
  ignoreGhostVars = FALSE,
  ignoreStrataVar = FALSE,
  randomizeRecords = "no"
)
```

Arguments

obj	object of class sdcMicroObj-class
ignoreKeyVars	If manipulated KeyVariables should be returned or the unchanged original variables
ignorePramVars	if manipulated PramVariables should be returned or the unchanged original variables
ignoreNumVars	if manipulated NumericVariables should be returned or the unchanged original variables
ignoreGhostVars	if manipulated Ghost (linked) Variables should be returned or the unchanged original variables
ignoreStrataVar	if manipulated StrataVariables should be returned or the unchanged original variables
randomizeRecords	(logical) specifies, if the output records should be randomized. The following options are possible:

'no' default, no randomization takes place

'simple' records are just randomly swapped.

'byHH' if slot 'hhId' is not NULL, the clusters defined by this variable are randomized across the dataset. If slot 'hhId' is NULL, the records or the dataset are randomly changed.

'withinHH' if slot 'hhId' is not NULL, the clusters defined by this variable are randomized across the dataset and additionally, the order of records within the clusters are also randomly changed. If slot 'hhId' is NULL, the records or the dataset are randomly changed.

Value

a data.frame containing the anonymized data set

Author(s)

Alexander Kowarik, Bernhard Meindl

Examples

```
## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- removeDirectID(sdc, var="age")
dataM <- extractManipData(sdc)
```

francdat *data from the casc project*

Description

Small synthetic data from Capobianchi, Polettini, Lucarelli

Format

A data frame with 8 observations on the following 8 variables.

Num1 a numeric vector

Key1 Key variable 1. A numeric vector

Num2 a numeric vector

Key2 Key variable 2. A numeric vector

Key3 Key variable 3. A numeric vector

Key4 Key variable 4. A numeric vector

Num3 a numeric vector

w The weight vector. A numeric vector

Details

This data set is very similar to that one which are used by the authors of the paper given below. We need this data set only for demonstration effect, i.e. that the package provides the same results as their software.

Source

https://research.cbs.nl/casc/deliv/12d1.pdf

Examples

```
data(francdat)
francdat
```

free1 *Demo data set from mu-Argus*

Description

The public use toy demo data set from the mu-Argus software for SDC.

Format

The format is: num [1:4000, 1:34] 36 36 36 36 36 36 36 36 36 36 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:34] "REGION" "SEX" "AGE" "MARSTAT" ...

Details

Please, see at the link given below. Please note, that the correlation structure of the data is not very realistic, especially concerning the continuous scaled variables which drawn independently from are a multivariate uniform distribution.

Source

Public use file from the CASC project.

Examples

```
data(free1)
head(free1)
```

freq *Freq*

Description

Extract sample frequency counts (fk) or estimated population frequency counts (Fk)

Usage

```
freq(obj, type = "fk")
```

Arguments

obj	an sdcMicroObj-class-object
type	either 'fk' or 'FK'

Value

a vector containing sample frequencies or weighted frequencies

Author(s)

Bernhard Meindl

Examples

```
data(testdata)
sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof','walls','relat','sex'),
  pramVars=c('water','electcon'),
  numVars=c('expend','income','savings'), w='sampling_weight')
head(freq(sdc, type="fk"))
head(freq(sdc, type="Fk"))
```

---

freqCalc *Frequencies calculation for risk estimation*

Description

Computation and estimation of the sample and population frequency counts.

Usage

```
freqCalc(x, keyVars, w = NULL, alpha = 1)
```

Arguments

x	data frame or matrix
keyVars	key variables
w	column index of the weight variable. Should be set to NULL if one deal with a population.
alpha	numeric value between 0 and 1 specifying how much keys that contain missing values (NAs) should contribute to the calculation of fk and Fk. For the default value of 1, nothing changes with respect to the implementation in prior versions. Each *wildcard-match* would be counted while for alpha=0 keys with missing values would be basically ignored.

Details

The function considers the case of missing values in the data. A missing value stands for any of the possible categories of the variable considered. It is possible to apply this function to large data sets with many (catergorical) key variables, since the computation is done in C.

freqCalc() does not support sdcMicro S4 class objects.

Value

Object from class freqCalc.

freqCalc	data set
keyVars	variables used for frequency calculation
w	index of weight vector. NULL if you do not have a sample.
alpha	value of parameter alpha
fk	the frequency of equal observations in the key variables subset sample given for each observation.
Fk	estimated frequency in the population
n1	number of observations with fk=1
n2	number of observations with fk=2

Author(s)

Bernhard Meindl

References

look e.g. in https://research.cbs.nl/casc/deliv/12d1.pdf Templ, M. *Statistical Disclosure Control for Microdata Using the R-Package sdcMicro*, Transactions on Data Privacy, vol. 1, number 2, pp. 67-85, 2008. https://www.tdp.cat/issues/abs.a004a08.php

Templ, M. *New Developments in Statistical Disclosure Control and Imputation: Robust Statistics Applied to Official Statistics*, Suedwestdeutscher Verlag fuer Hochschulschriften, 2009, ISBN: 3838108280, 264 pages.

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4 doi:10.1007/9783319502724

Templ, M. and Meindl, B.: *Practical Applications in Statistical Disclosure Control Using R*, Privacy and Anonymity in Information Management Systems New Techniques for New Practical Problems, Springer, 31-62, 2010, ISBN: 978-1-84996-237-7.

See Also

indivRisk, measure_risk

Examples

```
data(francdat)

f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$freqCalc
f$fk
f$Fk
## with missings:
x <- francdat
```

```
x[3,5] <- NA
x[4,2] <- x[4,4] <- NA
x[5,6]  <- NA
x[6,2]  <- NA
f2 <- freqCalc(x, keyVars=c(2,4,5,6),w=8)
cbind(f2$fk, f2$Fk)

## test parameter 'alpha'
f3a <- freqCalc(x, keyVars=c(2,4,5,6), w=8, alpha=1)
f3b <- freqCalc(x, keyVars=c(2,4,5,6), w=8, alpha=0.5)
f3c <- freqCalc(x, keyVars=c(2,4,5,6), w=8, alpha=0.1)
data.frame(fka=f3a$fk, fkb=f3b$fk, fkc=f3c$fk)
data.frame(Fka=f3a$Fk, Fkb=f3b$Fk, Fkc=f3c$Fk)
```

`generateStrata` *Generate one strata variable from multiple factors*

Description

For strata defined by multiple variables (e.g. sex,age,country) one combined variable is generated.

Usage

```
generateStrata(df, stratavars, name)
```

Arguments

`df`	a data.frame
`stratavars`	character vector with variable name
`name`	name of the newly generated variable

Value

The original data set with one new column.

Author(s)

Alexander Kowarik

Examples

```
x <- testdata
x <- generateStrata(x,c("sex","urbrur"),"strataIDvar")
head(x)
```

get.sdcMicroObj *get.sdcMicroObj*

Description

extract information from sdcMicroObj-class-objects depending on argument type

Usage

```
get.sdcMicroObj(object, type)
```

Arguments

object	a sdcMicroObj-class-object
type	a character vector of length 1 defining what to calculate|return|modify. Allowed types are are all slotNames of obj.

Value

a slot of a sdcMicroObj-class-object depending on argument type

Examples

```
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sl <- slotNames(sdc)
res <- sapply(sl, function(x) get.sdcMicroObj(sdc, type=x))
str(res)
```

globalRecode *Global Recoding*

Description

Global recoding of variables

Usage

```
globalRecode(obj, ...)
```

Arguments

obj	a numeric vector, a data.frame or an object of class sdcMicroObj-class
...	see possible arguments below

column: which keyVar should be changed. Character vector of length 1 specifying the variable name that should be recoded (required if obj is a data.frame or an object of class sdcMicroObj-class.

breaks: either a numeric vector of cut points or number giving the number of intervals which x is to be cut into.

labels: labels for the levels of the resulting category. By default, labels are constructed using "(a,b]" interval notation. If labels = FALSE, simple integer codes are returned instead of a factor.

method: The following arguments are supported:

- "equidistant:" for equal sized intervalls
- "logEqui:" for equal sized intervalls for log-transformed data
- "equalAmount:" for intervalls with approxiomately the same amount of observations

Details

If a labels parameter is specified, its values are used to name the factor levels. If none is specified, the factor level labels are constructed.

Value

the modified sdcMicroObj-class or a factor, unless labels = FALSE which results in the mere integer level codes.

Note

globalRecode can not be applied to vectors stored as factors from sdcMicro >= 4.7.0!

Author(s)

Matthias Templ and Bernhard Meindl

References

Templ, M. and Kowarik, A. and Meindl, B. Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro. *Journal of Statistical Software*, **67** (4), 1–36, 2015. doi:10.18637/jss.v067.i04

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4 doi:10.1007/9783319502724

See Also

cut

Examples

```
data(free1)
free1 <- as.data.frame(free1)

## application to a vector
head(globalRecode(free1$AGE, breaks=c(1,9,19,29,39,49,59,69,100), labels=1:8))
table(globalRecode(free1$AGE, breaks=c(1,9,19,29,39,49,59,69,100), labels=1:8))

## application to a data.frame
# automatic labels
table(globalRecode(free1, column="AGE", breaks=c(1,9,19,29,39,49,59,69,100))$AGE)

## calculation of brea-points using different algorithms
table(globalRecode(free1$AGE, breaks=6))
table(globalRecode(free1$AGE, breaks=6, method="logEqui"))
table(globalRecode(free1$AGE, breaks=6, method="equalAmount"))

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- globalRecode(sdc, column="water", breaks=3)
table(get.sdcMicroObj(sdc, type="manipKeyVars")$water)
```

`groupAndRename` *Join levels of a variables in an object of class* `sdcMicroObj-class` *or* `factor` *or* `data.frame`

Description

If the input is an object of class `sdcMicroObj-class`, the specified factor-variable is recoded into a factor with less levels and risk-measures are automatically recomputed.

Usage

```
groupAndRename(obj, var, before, after, addNA = FALSE)
```

Arguments

`obj`	object of class `sdcMicroObj-class`
`var`	name of the keyVariable to change
`before`	vector of levels before recoding
`after`	name of new level after recoding
`addNA`	logical, if TRUE missing values in the input variables are added to the level specified in argument `after`.

Details

If the input is of class `data.frame`, the result is a `data.frame` with a modified column specified by `var`.

If the input is of class `factor`, the result is a `factor` with different levels.

Value

the modified `sdcMicroObj-class`

Author(s)

Bernhard Meindl

References

Templ, M. and Kowarik, A. and Meindl, B. Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro. *Journal of Statistical Software*, **67** (4), 1–36, 2015. doi:10.18637/jss.v067.i04

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4 doi:10.1007/9783319502724

Examples

```
## for objects of class sdcMicro:
data(testdata2)
testdata2$urbrur <- as.factor(testdata2$urbrur)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- groupAndRename(sdc, var="urbrur", before=c("1","2"), after=c("1"))
```

IL_correl *Additional Information-Loss measures*

Description

Measures `IL_correl()` and `IL_variables()` were proposed by Andrzej Mlodak and are (theoretically) bounded between `0` and `1`.

Usage

```
IL_correl(x, xm)

## S3 method for class 'il_correl'
print(x, digits = 3, ...)

IL_variables(x, xm)
```

```
## S3 method for class 'il_variables'
print(x, digits = 3, ...)
```

Arguments

x	an object coercible to a data.frame representing the original dataset
xm	an object coercible to a data.frame representing the perturbed, modified dataset
digits	number digits used for rounding when displaying results
...	additional parameter for print-methods; currently ignored

Details

- IL_correl(): is a information-loss measure that can be applied to common numerically scaled variables in x and xm. It is based on diagonal entries of inverse correlation matrices in the original and perturbed data.
- IL_variables(): for common-variables in x and xm the individual distance-functions depend on the class of the variable; specifically these functions are different for numeric variables, ordered-factors and character/factor variables. The individual distances are summed up and scaled by n * m with n being the number of records and m being the number of (common) variables.

Details can be found in the references below

The implementation of IL_correl() differs slightly with the original proposition from Mlodak, A. (2020) as the constant multiplier was changed to 1 / sqrt(2) instead of 1/2 for better efficiency and interpretability of the measure.

Value

the corresponding information-loss measure

Author(s)

Bernhard Meindl bernhard.meindl@statistik.gv.at

References

Mlodak, A. (2020). Information loss resulting from statistical disclosure control of output data, Wiadomosci Statystyczne. The Polish Statistician, 2020, 65(9), 7-27, DOI: 10.5604/01.3001.0014.4121

Mlodak, A. (2019). Using the Complex Measure in an Assessment of the Information Loss Due to the Microdata Disclosure Control, Przegląd Statystyczny, 2019, 66(1), 7-26, DOI: 10.5604/01.3001.0013.8285

Examples

```
data("Tarragona", package = "sdcMicro")
res1 <- addNoise(obj = Tarragona, variables = colnames(Tarragona), noise = 100)
IL_correl(x = as.data.frame(res1$x), xm = as.data.frame(res1$xm))

res2 <- addNoise(obj = Tarragona, variables = colnames(Tarragona), noise = 25)
```

```
IL_correl(x = as.data.frame(res2$x), xm = as.data.frame(res2$xm))

# creating test-inputs
n <- 150
x <- xm <- data.frame(
  v1 = factor(sample(letters[1:5], n, replace = TRUE), levels = letters[1:5]),
  v2 = rnorm(n),
  v3 = runif(3),
  v4 = ordered(sample(LETTERS[1:3], n, replace = TRUE), levels = c("A", "B", "C"))
)
xm$v1[1:5] <- "a"
xm$v2 <- rnorm(n, mean = 5)
xm$v4[1:5] <- "A"
IL_variables(x, xm)
```

`importProblem` *importProblem*

Description

reads an sdcProblem with code that has been exported within `sdcApp`.

Usage

```
importProblem(path)
```

Arguments

`path`	a file path

Value

an object of class `sdcMicro_GUI_export` or an object of class 'simple.error'

Author(s)

Bernhard Meindl

`indivRisk` *Individual Risk computation*

Description

Estimation of the risk for each observation. After the risk is computed one can use e.g. the function localSuppr() for the protection of values of high risk. Further details can be found at the link given below.

Usage

```
indivRisk(x, method = "approx", qual = 1, survey = TRUE)
```

Arguments

`x`	object from class freqCalc
`method`	approx (default) or exact
`qual`	final correction factor
`survey`	TRUE, if we have survey data and FALSE if we deal with a population.

Details

S4 class sdcMicro objects are only supported by function *measure_risk* that also estimates the individual risk with the same method.

Value

rk: base individual risk

method: method

qual: final correction factor

fk: frequency count

knames: colnames of the key variables

Note

The base individual risk method was developed by Benedetti, Capobianchi and Franconi

Author(s)

Matthias Templ. Bug in method “exact” fixed since version 2.6.5. by Youri Baeyens.

References

Templ, M. and Kowarik, A. and Meindl, B. Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro. *Journal of Statistical Software*, **67** (4), 1–36, 2015. doi:10.18637/jss.v067.i04

Franconi, L. and Polettini, S. (2004) *Individual risk estimation in mu-Argus: a review*. Privacy in Statistical Databases, Lecture Notes in Computer Science, 262–272. Springer

Machanavajjhala, A. and Kifer, D. and Gehrke, J. and Venkitasubramaniam, M. (2007) *l-Diversity: Privacy Beyond k-Anonymity*. ACM Trans. Knowl. Discov. Data, 1(1)

additionally, have a look at the vignettes of sdcMicro for further reading.

See Also

measure_risk, freqCalc

Examples

```
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$fk
f$Fk
## individual risk calculation:
indivf <- indivRisk(f)
indivf$rk
```

---

infoLoss *Calculate information loss after targeted record swapping*

Description

Calculate information loss after targeted record swapping using both the original and the swapped micro data. Information loss will be calculated on table counts defined by parameter 'table_vars' using either implemented information loss measures like absolute deviaton, relative absolute deviation and absolute deviation of square roots or custom metric, See details below.

Usage

```
infoLoss(
  data,
  data_swapped,
  table_vars,
  metric = c("absD", "relabsD", "abssqrtD"),
  custom_metric = NULL,
  hid = NULL,
  probs = sort(c(seq(0, 1, by = 0.1), 0.95, 0.99)),
  quantvals = c(0, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, Inf),
```

```
  apply_quantvals = c("relabsD", "abssqrtD"),
  exclude_zeros = FALSE,
  only_inner_cells = FALSE
)
```

Arguments

data	original micro data set, must be either a ‘data.table‘ or ‘data.frame‘.
data_swapped	micro data set after targeted record swapping was applied. Must be either a ‘data.table‘ or ‘data.frame‘.
table_vars	column names in both ‘data‘ and ‘data_swapped‘. Defines the variables over which a (multidimensional) frequency table is constructed. Information loss is then calculated by applying the metric in ‘metric‘ and ‘custom_merics‘ over the cell-counts and margin counts of the table from ‘data‘ and ‘data_swapped‘.
metric	character vector containing one or more of the already implemented metrices: "absD","relabsD" and/or "abssqrtD".
custom_metric	function or (named) list of functions. Functions defined here must be of the form ‘fun(x,y,...)‘ where ‘x‘ and ‘y‘ expect numeric values of the same length. The output of these functions must be a numeric vector of the same length as ‘x‘ and ‘y‘.
hid	‘NULL‘ or character containing household id in ‘data‘ and ‘data_swapped‘. If not ‘NULL‘ frequencies will reflect number of households, otherwise frequencies will reflect number of persons.
probs	numeric vector containing values in the inervall [0,1].
quantvals	optional numeric vector which defines the groups used for the cumulative outputs. Is applied on the results ‘m‘ from each information loss metric as ‘cut(m,breaks=quantvals,include.lo see also return values.
apply_quantvals	character vector defining for the output of which metrices ‘quantvals‘ should be applied to.
exclude_zeros	‘TRUE‘ or ‘FALSE‘, if ‘TRUE‘ 0 cells in the frequency table using ‘data_swapped‘ will be ignored.
only_inner_cells	‘TRUE‘ or ‘FALSE‘, if ‘TRUE‘ only inner cells of the frequency table defined by ‘table_vars‘ will be compared. Otherwise also all tables margins will bei calculated.

Details

First frequency tables are build from both ‘data‘ and ‘data_swapped‘ using the variables defined in ‘table_vars‘. By default also all table margins will be calculated, see parameter ‘only_inner_cells = FALSE‘. After that the information loss metrices defined in either ‘metric‘ or ‘custom_metric‘ are applied on each of the table cells from both frequency tables. This is done in the sense of ‘metric(x,y)‘ where ‘metric‘ is the information loss, ‘x‘ a cell from the table created from ‘data‘ and ‘y‘ the same cell from the table created from ‘data_swapped‘. One or more custom metrices can be applied using the parameter ‘custom_metric‘, see also examples.

Value

Returns a list containing:

* ‘cellvalues‘: ‘data.table‘ showing in a long format for each table cell the frequency counts for ‘data‘ ~ ‘count_o‘ and ‘data_swapped‘ ~ ‘count_s‘. * ‘overview‘: ‘data.table‘ containing the disribution of the ‘noise‘ in number of cells and percentage. The ‘noise‘ ist calculated as the difference between the cell values of the frequency table generated from the original and swapped data * ‘measures‘: ‘data.table‘ containing the quantiles and mean (column ‘waht‘) of the distribution of the information loss metrices applied on each table cell. The quantiles are defined by parameter ‘probs‘. * ‘cumdistr*‘: ‘data.table‘ containing the cumulative distribution of the information loss metrices. Distribution is shown in number of cells (‘cnt‘) and percentage (‘pct‘). Column ‘cat‘ shows all unique values of the information loss metric or the grouping defined by ‘quantvals‘. * ‘false_zero‘: number of table cells which are non-zero when using ‘data‘ and zero when using ‘data_swapped‘. * ‘false_nonzero‘: number of table cells which are zero when using ‘data‘ and non-zero when using ‘data_swapped‘. * ‘exclude_zeros‘: value passed to ‘exclude_zero‘ when calling the function.

Examples

```
# generate dummy data
seed <- 2021
set.seed(seed)
nhid <- 10000
dat <- createDat( nhid )

# define paramters for swapping
k_anonymity <- 1
swaprate <- .05
similar <- list(c("hsize"))
hier <- c("nuts1","nuts2")
carry_along <- c("nuts3","lau2")
risk_variables <- c("ageGroup","national")
hid <- "hid"

# # apply record swapping
# dat_s <- recordSwap(data = dat, hid = hid, hierarchy = hier,
#                     similar = similar, swaprate = swaprate,
#                     k_anonymity = k_anonymity,
#                     risk_variables = risk_variables,
#                     carry_along = carry_along,
#                     return_swapped_id = TRUE,
#                     seed=seed)
#
#
# # calculate informationn loss
# # for the table nuts2 x national
# iloss <- infoLoss(data=dat, data_swapped = dat_s,
#                   table_vars = c("nuts2","national"))
# iloss$measures # distribution of information loss measures
# iloss$false_zero # no false zeros
# iloss$false_nonzero # no false non-zeros
#
```

```
# # frequency tables of households accross
# # nuts2 x hincome
#
# iloss <- infoLoss(data=dat, data_swapped = dat_s,
 #                  table_vars = c("nuts2","hincome"),
#                   hid = "hid")
# iloss$measures
#
# # define custom metric
# squareD <- function(x,y){
#   (x-y)^2
# }
#
# iloss <- infoLoss(data=dat, data_swapped = dat_s,
#                   table_vars = c("nuts2","national"),
#                   custom_metric = list(squareD=squareD))
# iloss$measures # includes custom loss as well
#
```

---

kAnon_violations kAnon_violations

Description

returns the number of observations violating k-anonymity.

Usage

```
kAnon_violations(object, weighted, k)

## S4 method for signature 'sdcMicroObj,logical,numeric'
kAnon_violations(object, weighted, k)
```

Arguments

object	a sdcMicroObj-class object
weighted	TRUE or FALSE defining if sampling weights should be taken into account
k	a positive number defining parameter k

Value

the number of records that are violating k-anonymity based on unweighted sample data only (in case parameter weighted is FALSE) or computing the number of observations that are estimated to violate k-anonymity in the population in case parameter weighted equals TRUE.

LocalRecProg *Local recoding via Edmond's maximum weighted matching algorithm*

Description

To be used on both categorical and numeric input variables, although usage on categorical variables is the focus of the development of this software.

Usage

```
LocalRecProg(
  obj,
  ancestors = NULL,
  ancestor_setting = NULL,
  k_level = 2,
  FindLowestK = TRUE,
  weight = NULL,
  lowMemory = FALSE,
  missingValue = NA,
  ...
)
```

Arguments

obj	a data.frame or a sdcMicroObj-class-object
ancestors	Names of ancestors of the cateorical variables
ancestor_setting	For each ancestor the corresponding categorical variable
k_level	Level for k-anonymity
FindLowestK	requests the program to look for the smallest k that results in complete matches of the data.
weight	A weight for each variable (Default=1)
lowMemory	Slower algorithm with less memory consumption
missingValue	The output value for a suppressed value.
...	see arguments below
	categorical Names of categorical variables
	numerical Names of numerical variables

Details

Each record in the data represents a category of the original data, and hence all records in the input data should be unique by the N Input Variables. To achieve bigger category sizes (k-anoymity), one can form new categories based on the recoding result and repeatedly apply this algorithm.

Value

dataframe with original variables and the supressed variables (suffix _lr). / the modified sdcMicroObj-class

Methods

list("signature(obj=\"sdcMicroObj\")")

Author(s)

Alexander Kowarik, Bernd Prantner, IHSN C++ source, Akimichi Takemura

References

Kowarik, A. and Templ, M. and Meindl, B. and Fonteneau, F. and Prantner, B.: *Testing of IHSN Cpp Code and Inclusion of New Methods into sdcMicro*, in: Lecture Notes in Computer Science, J. Domingo-Ferrer, I. Tinnirello (editors.); Springer, Berlin, 2012, ISBN: 978-3-642-33626-3, pp. 63-77. doi:10.1007/9783642336270_6

Examples

```
data(testdata2)
cat_vars <- c("urbrur", "roof", "walls", "water", "sex", "relat")
anc_var <- c("water2", "water3", "relat2")
anc_setting <- c("water","water","relat")

r1 <- LocalRecProg(
  obj = testdata2,
  categorical = cat_vars,
  missingValue = -99)
r2 <- LocalRecProg(
  obj = testdata2,
  categorical = cat_vars,
  ancestor = anc_var,
  ancestor_setting = anc_setting,
  missingValue = -99)
r3 <- LocalRecProg(
  obj = testdata2,
  categorical = cat_vars,
  ancestor = anc_var,
  ancestor_setting = anc_setting,
  missingValue = -99,
  FindLowestK = FALSE)

# for objects of class sdcMicro:
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = c("urbrur", "roof", "walls", "water", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight")
sdc <- LocalRecProg(sdc)
```

localSupp *Local Suppression*

Description

A simple method to perfom local suppression.

Usage

```
localSupp(obj, threshold = 0.15, keyVar)
```

Arguments

obj	object of class freqCalc or sdcMicroObj-class.
threshold	threshold for individual risk
keyVar	Variable on which some values might be suppressed

Details

Values of high risk (above the threshold) of a certain variable (parameter keyVar) are suppressed.

Value

an updated object of class freqCalc or the sdcMicroObj-class object with manipulated data.

Author(s)

Matthias Templ and Bernhard Meindl

References

Templ, M. *Statistical Disclosure Control for Microdata Using the R-Package sdcMicro*, Transactions on Data Privacy, vol. 1, number 2, pp. 67-85, 2008. http://www.tdp.cat/issues/abs.a004a08.php

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4 doi:10.1007/9783319502724

See Also

freqCalc, indivRisk

Examples

```
data(francdat)
keyVars <- paste0("Key",1:4)

f <- freqCalc(francdat, keyVars = keyVars, w = 8)
f
f$fk
f$Fk

## individual risk calculation:
indivf <- indivRisk(f)
indivf$rk

## Local Suppression
localS <- localSupp(f, keyVar = "Key4", threshold = 0.15)
f2 <- freqCalc(localS$freqCalc, keyVars = keyVars, w = 8)
indivf2 <- indivRisk(f2)
indivf2$rk
identical(indivf$rk, indivf2$rk)

## select another keyVar and run localSupp once again,
# if you think the table is not fully protected

## for objects of class sdcMicro:
data(testdata)
sdc <- createSdcObj(
  dat = testdata,
  keyVars = c("urbrur", "roof", "walls", "water", "electcon", "relat", "sex"),
  w = "sampling_weight"
)
sdc <- localSupp(sdc, keyVar = "urbrur", threshold = 0.045)
print(sdc, type = "ls")
```

---

localSuppression *Local Suppression to obtain k-anonymity*

Description

Algorithm to achieve *k*-anonymity by performing local suppression.

Usage

```
localSuppression(obj, k = 2, importance = NULL, combs = NULL, ...)

kAnon(obj, k = 2, importance = NULL, combs = NULL, ...)
```

Arguments

obj	a sdcMicroObj-class object or a data.frame
k	Threshold for *k*-anonymity
importance	Numeric vector of values between 1 and *n* (n = length(keyVars)). This vector defines the "importance" of variables for local suppression. Variables with importance = 1 will, if possible, not be suppressed; variables with importance = n will be prioritized for suppression.
combs	Numeric vector. If specified, the algorithm provides *k*-anonymity for each combination of *n* key variables (with *n* being the value of the ith element of this parameter). For example, combs = c(4,3) means that *k*-anonymity will be provided for all combinations of 4 and then 3 key variables. It is possible to assign different *k* values for each combination by supplying k as a vector. If k has only one value, it will be used for all subsets.
...	see additional arguments below: • keyVars: Names or indices of categorical key variables (for data.frame method) • strataVars: Name or index of the variable used for stratification. *k*-anonymity is ensured within each category of this variable. • alpha: Numeric value between 0 and 1 specifying how much keys with missing values (NAs) contribute to the calculation of fk and Fk. Default is 1. Used only in the data.frame method. • nc: Maximum number of cores used for stratified computations. Default is 1. Parallelization is ignored on Windows.

Details

The algorithm provides a *k*-anonymized data set by suppressing values in key variables. The algorithm tries to find an optimal solution to suppress as few values as possible and considers the specified importance vector. If not specified, the importance vector is constructed in a way such that key variables with a high number of characteristics are considered less important than key variables with a low number of characteristics.

The implementation provides *k*-anonymity per strata, if slot strataVar has been set in sdcMicroObj-class or if parameter strataVar is used when applying the data.frame method. For details, see the examples provided.

For the parameter alpha:

- alpha = 1 counts all *wildcard matches* (i.e. NAs match everything).
- alpha = 0 assumes missing values form their own categories.

These are two extremes. With alpha = 0, frequencies are likely underestimated when NAs are present. If combs is used with alpha = 0, the heuristic nature of kAnon() may lead to technically correct, but not always intuitively understandable frequency evaluations.

Value

A modified dataset with suppressions that meets *k*-anonymity based on the specified key variables, or the modified sdcMicroObj-class object.

Note

Deprecated methods `localSupp2` and `localSupp2Wrapper` are no longer available in `sdcMicro` versions > 4.5.0. `kAnon()` is a more intuitive term for local suppression, since the goal is to achieve *k*-anonymity.

Author(s)

Bernhard Meindl, Matthias Templ

References

Templ, M. *Statistical Disclosure Control for Microdata: Methods and Applications in R.* Springer International Publishing, 287 pages, 2017. ISBN: 978-3-319-50272-4. doi:10.1007/978331950272-4

Templ, M., Kowarik, A., Meindl, B. *Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro.* Journal of Statistical Software, **67**(4), 1–36, 2015. doi:10.18637/jss.v067.i04

Examples

```
data(francdat)

## Local Suppression
localS <- localSuppression(francdat, keyVar = c(4, 5, 6))
localS
plot(localS)

## for objects of class sdcMicro, no stratification
data(testdata2)
kv <- c("urbrur", "roof", "walls", "water", "electcon", "relat", "sex")
sdc <- createSdcObj(testdata2, keyVars = kv, w = "sampling_weight")
sdc <- localSuppression(sdc)

## for objects of class sdcMicro, with stratification
testdata2$ageG <- cut(testdata2$age, 5, labels = paste0("AG", 1:5))
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = kv,
  w = "sampling_weight",
  strataVar = "ageG"
)
sdc <- localSuppression(sdc, nc = 1)

## it is also possible to provide k-anonymity for subsets of key-variables
## with different parameter k!
## in this case we want to provide 10-anonymity for all combinations
## of 5 key variables, 20-anonymity for all combinations with 4 key variables
## and 30-anonymity for all combinations of 3 key variables.
sdc <- createSdcObj(testdata2, keyVars = kv, w = "sampling_weight")
combs <- 5:3
k <- c(10, 20, 30)
sdc <- localSuppression(sdc, k = k, combs = combs)
```

```
## data.frame method (no stratification)
inp <- testdata2[, c(kv, "ageG")]
ls <- localSuppression(inp, keyVars = 1:7)
print(ls)
plot(ls)

## data.frame method (with stratification)
ls <- kAnon(inp, keyVars = 1:7, strataVars = 8)
print(ls)
plot(ls)
```

`mafast` *Fast and Simple Microaggregation*

Description

Function to perform a fast and simple (primitive) method of microaggregation. (for large datasets)

Usage

```
mafast(obj, variables = NULL, by = NULL, aggr = 3, measure = mean)
```

Arguments

`obj`	either a `sdcMicroObj-class`-object or a `data.frame`
`variables`	variables to microaggregate. If obj is of class sdcMicroObj the numerical key variables are chosen per default.
`by`	grouping variable for microaggregation. If obj is of class sdcMicroObj the strata variables are chosen per default.
`aggr`	aggregation level (default=3)
`measure`	aggregation statistic, mean, median, trim, onestep (default = mean)

Value

If 'obj' was of class `sdcMicroObj-class` the corresponding slots are filled, like manipNumVars, risk and utility. If 'obj' was of class "data.frame" or "matrix" an object of the same class is returned.

Author(s)

Alexander Kowarik

See Also

`microaggregation`

Examples

```
data(Tarragona)
m1 <- mafast(Tarragona, variables=c("GROSS.PROFIT","OPERATING.PROFIT","SALES"),aggr=3)
data(testdata)
m2 <- mafast(testdata,variables=c("expend","income","savings"),aggr=50,by="sex")
summary(m2)

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- dRisk(sdc)
sdc@risk$numeric
sdc1 <- mafast(sdc,aggr=4)
sdc1@risk$numeric

sdc2 <- mafast(sdc,aggr=10)
sdc2@risk$numeric

### Performance tests
x <- testdata
for(i in 1:20){
  x <- rbind(x,testdata)
}
system.time({
  xx <- mafast(
    obj = x,
    variables = c("expend", "income", "savings"),
    aggr = 50,
    by = "sex"
  )
})
```

`measure_risk` *Disclosure Risk for Categorical Variables*

Description

The function measures the disclosure risk for weighted or unweighted data. It computes the individual risk (and household risk if reasonable) and the global risk. It also computes a risk threshold based on a global risk value.

Prints a 'measure_risk'-object

Prints a 'ldiversity'-object

Usage

```
measure_risk(obj, ...)

ldiversity(obj, ldiv_index = NULL, l_recurs_c = 2, missing = -999, ...)

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

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

Arguments

obj	Object of class sdcMicroObj-class
...	see arguments below
	data: Input data, a data.frame.
	keyVars: names (or indices) of categorical key variables (for data-frame method)
	w: name of variable containing sample weights
	hid: name of the clustering variable, e.g. the household ID
	max_global_risk: Maximal global risk for threshold computation
	fast_hier: If TRUE a fast approximation is computed if household data are provided.
ldiv_index	indices (or names) of the variables used for l-diversity
l_recurs_c	l-Diversity Constant
missing	a integer value to be used as missing value in the C++ routine
x	Output of measure_risk() or ldiversity()

Details

To be used when risk of disclosure for individuals within a family is considered to be statistical independent.

Internally, function *freqCalc()* and *indivRisk* are used for estimation.

Measuring individual risk: The individual risk approach based on so-called super-population models. In such models population frequency counts are modeled given a certain distribution. The estimation procedure of sample frequency counts given the population frequency counts is modeled by assuming a negative binomial distribution. This is used for the estimation of the individual risk. The extensive theory can be found in Skinner (1998), the approximation formulas for the individual risk used is described in Franconi and Polettini (2004).

Measuring hierarchical risk: If "hid" - the index of variable holding information on the hierarchical cluster structures (e.g., individuals that are clustered in households) - is provided, the hierarchical risk is additional estimated. Note that the risk of re-identifying an individual within a household may also affect the probability of disclosure of other members in the same household. Thus, the household or cluster-structure of the data must be taken into account when estimating disclosure risks. It is commonly assumed that the risk of re-identification of a household is the risk that at least

one member of the household can be disclosed. Thus this probability can be simply estimated from individual risks as 1 minus the probability that no member of the household can be identified.

Global risk: The sum of the individual risks in the dataset gives the expected number of re-identifications that serves as measure of the global risk.

l-Diversity: If "ldiv_index" is unequal to NULL, i.e. if the indices of sensible variables are specified, various measures for l-diversity are calculated. l-diverstiy is an extension of the well-known k-anonymity approach where also the uniqueness in sensible variables for each pattern spanned by the key variables are evaluated.

Value

A modified `sdcMicroObj-class` object or a list with the following elements:

global_risk_ER: expected number of re-identification.

global_risk: global risk (sum of indivdual risks).

global_risk_pct: global risk in percent.

Res: matrix with the risk, frequency in the sample and grossed-up frequency in the population (and the hierachical risk) for each observation.

global_threshold: for a given max_global_risk the threshold for the risk of observations.

max_global_risk: the input max_global_risk of the function.

hier_risk_ER: expected number of re-identification with household structure.

hier_risk: global risk with household structure (sum of indivdual risks).

hier_risk_pct: global risk with household structure in percent.

ldiverstiy: Matrix with Distinct_Ldiversity, Entropy_Ldiversity and Recursive_Ldiversity for each sensitivity variable.

Prints risk-information into the console

Information on L-Diversity Measures in the console

Author(s)

Alexander Kowarik, Bernhard Meindl, Matthias Templ, Bernd Prantner, minor parts of IHSN C++ source

References

Franconi, L. and Polettini, S. (2004) *Individual risk estimation in mu-Argus: a review*. Privacy in Statistical Databases, Lecture Notes in Computer Science, 262–272. Springer

Machanavajjhala, A. and Kifer, D. and Gehrke, J. and Venkitasubramaniam, M. (2007) *l-Diversity: Privacy Beyond k-Anonymity*. ACM Trans. Knowl. Discov. Data, 1(1)

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4.

#' Templ, M. and Kowarik, A. and Meindl, B. Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro. *Journal of Statistical Software*, **67** (4), 1–36, 2015. doi:10.18637/jss.v067.i04

See Also

freqCalc, indivRisk

measure_risk

Examples

```
## measure_risk with sdcMicro objects:
data(testdata)

sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof','walls','water','electcon'),
numVars=c('expend','income','savings'), w='sampling_weight')

## risk is already estimated and available in...
names(sdc@risk)

## measure risk on data frames or matrices:
res <- measure_risk(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"))
print(res)
head(res$Res)
resw <- measure_risk(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"),w="sampling_weight")
print(resw)
head(resw$Res)
res1 <- ldiversity(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"),ldiv_index="electcon")
print(res1)
head(res1)
res2 <- ldiversity(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"),ldiv_index=c("electcon","relat"))
print(res2)
head(res2)

# measure risk with household risk
resh <- measure_risk(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"),w="sampling_weight",hid="ori_hid")
print(resh)

# change max_global_risk
rest <- measure_risk(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"),
  w="sampling_weight",max_global_risk=0.0001)
print(rest)

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
## -> when using `createSdcObj()`, the risks are already internally computed
## and it is not required to explicitely run `sdc <- measure_risk(sdc)`
```

mergeHouseholdData	*Replaces the raw household-level data with the anonymized household-level data in the full dataset for anonymization of data with a household structure (or other hierarchical structure). Requires a matching household ID in both files.*

Description

Replaces the raw household-level data with the anonymized household-level data in the full dataset for anonymization of data with a household structure (or other hierarchical structure). Requires a matching household ID in both files.

Usage

```
mergeHouseholdData(dat, hhId, dathh)
```

Arguments

dat	a data.frame with the full dataset
hhId	name of the household (cluster) ID (identical in both datasets)
dathh	a dataframe with the treated household level data (generated for example with selectHouseholdData)

Value

a data.frame with the treated household level variables and the raw individual level variables

Author(s)

Thijs Benschop and Bernhard Meindl

Examples

```
## Load data
x <- testdata

## donttest is necessary because of
## Examples with CPU time > 2.5 times elapsed time
## caused by using C++ code and/or data.table
## Create household level dataset
x_hh <- selectHouseholdData(dat=x, hhId="ori_hid",
  hhVars=c("urbrur", "roof",  "walls", "water", "electcon", "household_weights"))
## Anonymize household level dataset and extract data
sdc_hh <- createSdcObj(x_hh, keyVars=c('urbrur','roof'), w='household_weights')
sdc_hh <- kAnon(sdc_hh, k = 3)
x_hh_anon <- extractManipData(sdc_hh)
```

```
## Merge anonymized household level data back into the full dataset
x_anonhh <- mergeHouseholdData(x, "ori_hid", x_hh_anon)

## Anonymize full dataset and extract data
sdc_full <- createSdcObj(x_anonhh, keyVars=c('sex', 'age', 'urbrur', 'roof'), w='sampling_weight')
sdc_full <- kAnon(sdc_full, k = 3)
x_full_anon <- extractManipData(sdc_full)
```

`microaggregation` *Microaggregation*

Description

Function to perform various methods of microaggregation.

Usage

```
microaggregation(
  obj,
  variables = NULL,
  aggr = 3,
  strata_variables = NULL,
  method = "mdav",
  weights = NULL,
  nc = 8,
  clustermethod = "clara",
  measure = "mean",
  trim = 0,
  varsort = 1,
  transf = "log"
)
```

Arguments

`obj`	either an object of class `sdcMicroObj-class` or a `data.frame`
`variables`	variables to microaggregate. For `NULL`: If obj is of class sdcMicroObj, all numerical key variables are chosen per default. For `data.frames`, all columns are chosen per default.
`aggr`	aggregation level (default=3)
`strata_variables`	for `data.frames`, by-variables for applying microaggregation only within strata defined by the variables. For `sdcMicroObj-class`-objects, the stratification-variable defined in slot `@strataVar` is used. This slot can be changed any time using `strataVar<-`.

method	pca, rmd, onedims, single, simple, clustpca, pppca, clustpppca, mdav, clustmcdpca, influence, mcdpca
weights	sampling weights. If obj is of class sdcMicroObj the vector of sampling weights is chosen automatically. If determined, a weighted version of the aggregation measure is chosen automatically, e.g. weighted median or weighted mean.
nc	number of cluster, if the chosen method performs cluster analysis
clustermethod	clustermethod, if necessary
measure	aggregation statistic, mean, median, trim, onestep (default=mean)
trim	trimming percentage, if measure=trim
varsort	variable for sorting, if method=single
transf	transformation for data x

Details

On https://research.cbs.nl/casc/glossary.htm one can found the "official" definition of microaggregation:

Records are grouped based on a proximity measure of variables of interest, and the same small groups of records are used in calculating aggregates for those variables. The aggregates are released instead of the individual record values.

The recommended method is "rmd" which forms the proximity using multivariate distances based on robust methods. It is an extension of the well-known method "mdav". However, when computational speed is important, method "mdav" is the preferable choice.

While for the proximity measure very different concepts can be used, the aggregation itself is naturally done with the arithmetic mean. Nevertheless, other measures of location can be used for aggregation, especially when the group size for aggregation has been taken higher than 3. Since the median seems to be unsuitable for microaggregation because of being highly robust, other mesures which are included can be chosen. If a complex sample survey is microaggregated, the corresponding sampling weights should be determined to either aggregate the values by the weighted arithmetic mean or the weighted median.

This function contains also a method with which the data can be clustered with a variety of different clustering algorithms. Clustering observations before applying microaggregation might be useful. Note, that the data are automatically standardised before clustering.

The usage of clustering method 'Mclust' requires package mclust02, which must be loaded first. The package is not loaded automatically, since the package is not under GPL but comes with a different licence.

The are also some projection methods for microaggregation included. The robust version 'pppca' or 'clustpppca' (clustering at first) are fast implementations and provide almost everytime the best results.

Univariate statistics are preserved best with the individual ranking method (we called them 'onedims', however, often this method is named 'individual ranking'), but multivariate statistics are strong affected.

With method 'simple' one can apply microaggregation directly on the (unsorted) data. It is useful for the comparison with other methods as a benchmark, i.e. replies the question how much better is a sorting of the data before aggregation.

Value

If 'obj' was of class sdcMicroObj-class the corresponding slots are filled, like manipNumVars, risk and utility. If 'obj' was of class "data.frame", an object of class "micro" with following entities is returned:

`x:` original data

`mx:` the microaggregated dataset

`method:` method

`aggr:` aggregation level

`measure:` proximity measure for aggregation

Note

if only one variable is specified, mafast is applied and argument method is ignored. Parameters measure are ignored for methods mdav and rmd.

Author(s)

Matthias Templ, Bernhard Meindl

For method "mdav": This work is being supported by the International Household Survey Network (IHSN) and funded by a DGF Grant provided by the World Bank to the PARIS21 Secretariat at the Organisation for Economic Co-operation and Development (OECD). This work builds on previous work which is elsewhere acknowledged.

Author for the integration of the code for mdav in R: Alexander Kowarik.

References

Templ, M. and Meindl, B., *Robust Statistics Meets SDC: New Disclosure Risk Measures for Continuous Microdata Masking*, Lecture Notes in Computer Science, Privacy in Statistical Databases, vol. 5262, pp. 113-126, 2008.

Templ, M. *Statistical Disclosure Control for Microdata Using the R-Package sdcMicro*, Transactions on Data Privacy, vol. 1, number 2, pp. 67-85, 2008. http://www.tdp.cat/issues/abs.a004a08.php

Templ, M. *New Developments in Statistical Disclosure Control and Imputation: Robust Statistics Applied to Official Statistics*, Suedwestdeutscher Verlag fuer Hochschulschriften, 2009, ISBN: 3838108280, 264 pages.

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4 doi:10.1007/9783319502724

Templ, M. and Meindl, B. and Kowarik, A.: *Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro*, Journal of Statistical Software, 67 (4), 1–36, 2015.

See Also

summary.micro, plotMicro, valTable

Examples

```
data(testdata)
# donttest since Examples with CPU time larger 2.5 times elapsed time, because
# of using data.table and multicore computation.

m <- microaggregation(
  obj = testdata[1:100, c("expend", "income", "savings")],
  method = "mdav",
  aggr = 4
)
summary(m)

## for objects of class sdcMicro:
## no stratification because `@strataVar` is `NULL`
data(testdata2)
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = c("urbrur", "roof", "walls", "water", "electcon", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight"
)
sdc <- microaggregation(
  obj = sdc,
  variables = c("expend", "income")
)

## with stratification using variable `"relat"`
strataVar(sdc) <- "relat"
sdc <- microaggregation(
  obj = sdc,
  variables = "savings"
)
```

`microaggrGower` *Microaggregation for numerical and categorical key variables based on a distance similar to the Gower Distance*

Description

The microaggregation is based on the distances computed similar to the Gower distance. The distance function makes distinction between the variable types factor,ordered,numerical and mixed (semi-continuous variables with a fixed probability mass at a constant value e.g. 0)

Usage

```
microaggrGower(
  obj,
  variables = NULL,
```

```
  aggr = 3,
  dist_var = NULL,
  by = NULL,
  mixed = NULL,
  mixed.constant = NULL,
  trace = FALSE,
  weights = NULL,
  numFun = mean,
  catFun = VIM::sampleCat,
  addRandom = FALSE
)
```

Arguments

obj	sdcMicroObj-class-object or a data.frame
variables	character vector with names of variables to be aggregated (Default for sdcMicroObj is all keyVariables and all numeric key variables)
aggr	aggregation level (default=3)
dist_var	character vector with variable names for distance computation
by	character vector with variable names to split the dataset before performing microaggregation (Default for sdcMicroObj is strataVar)
mixed	character vector with names of mixed variables
mixed.constant	numeric vector with length equal to mixed, where the mixed variables have the probability mass
trace	TRUE/FALSE for some console output
weights	numerical vector with length equal the number of variables for distance computation
numFun	function: to be used to aggregated numerical variables
catFun	function: to be used to aggregated categorical variables
addRandom	TRUE/FALSE if a random value should be added for the distance computation.

Details

The function sampleCat samples with probabilities corresponding to the occurrence of the level in the NNs. The function maxCat chooses the level with the most occurrences and random if the maximum is not unique.

Value

The function returns the updated sdcMicroObj or simply an altered data frame.

Note

In each by group all distance are computed, therefore introducing more by-groups significantly decreases the computation time and memory consumption.

Author(s)

Alexander Kowarik

See Also

sampleCat and maxCat

Examples

```
data(testdata,package="sdcMicro")
testdata <- testdata[1:200,]

for(i in c(1:7,9)) testdata[,i] <- as.factor(testdata[,i])
test <- microaggrGower(testdata,variables=c("relat","age","expend"),
  dist_var=c("age","sex","income","savings"),by=c("urbrur","roof"))

for(i in c(1:7,9)) testdata[,i] <- as.ordered(testdata[,i])
sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')

sdc <- microaggrGower(sdc)
```

---

microData *microData*

Description

Small aritificial toy data set.

Format

The format is: num [1:13, 1:5] 5 7 2 1 7 8 12 3 15 4 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:13] "10000" "11000" "12000" "12100" ... ..$: chr [1:5] "one" "two" "three" "four" ...

Examples

```
data(microData)
microData <- as.data.frame(microData)
m1 <- microaggregation(microData, method="mdav")
summary(m1)
```

modRisk *Global risk using log-linear models.*

Description

The sample frequencies are assumed to be independent and following a Poisson distribution. The parameters of the corresponding parameters are estimated by a log-linear model including the main effects and possible interactions.

Usage

```
modRisk(obj, method = "default", weights, formulaM, bound = Inf, ...)
```

Arguments

obj	An sdcMicroObj-class-object or a numeric matrix or data.frame containing all variables required in the specified model.
method	chose method for model-based risk-estimation. Currently, the following methods can be selected: • "default": the standard log-linear model. • "CE": the Clogg Eliason method, additionally, considers survey weights by using an offset term. • "PML": the pseudo maximum likelihood method. • "weightedLLM": the weighted maximum likelihood method, considers survey weights by including them as one of the predictors. • "IPF": iterative proportional fitting as used in deprecated method 'LLmodGlobalRisk'.
weights	a variable name specifying sampling weights
formulaM	A formula specifying the model.
bound	a number specifying a threshold for 'risky' observations in the sample.
...	additional parameters passed through, currently ignored.

Details

This measure aims to (1) calculate the number of sample uniques that are population uniques with a probabilistic Poisson model and (2) to estimate the expected number of correct matches for sample uniques.

ad 1) this risk measure is defined over all sample uniques as

$$\tau_1 = \sum_{j:f_j=1} P(F_j = 1 | f_j = 1) \quad ,$$

i.e. the expected number of sample uniques that are population uniques.

ad 2) this risk measure is defined over all sample uniques as

$$\tau_2 = \sum_{j:f_j=1} P(1/F_j|f_j = 1) \quad .$$

Since population frequencies F_k are unknown, they need to be estimated.

The iterative proportional fitting method is used to fit the parameters of the Poisson distributed frequency counts related to the model specified to fit the frequency counts. The obtained parameters are used to estimate a global risk, defined in Skinner and Holmes (1998).

Value

Two global risk measures and some model output given the specified model. If this method is applied to an sdcMicroObj-class-object, the slot 'risk' in the object ist updated with the result of the model-based risk-calculation.

Author(s)

Matthias Templ, Marius Totter, Bernhard Meindl

References

Skinner, C.J. and Holmes, D.J. (1998) *Estimating the re-identification risk per record in microdata*. Journal of Official Statistics, 14:361-372, 1998.

Rinott, Y. and Shlomo, N. (1998). *A Generalized Negative Binomial Smoothing Model for Sample Disclosure Risk Estimation*. Privacy in Statistical Databases. Lecture Notes in Computer Science. Springer-Verlag, 82–93.

Clogg, C.C. and Eliasson, S.R. (1987). *Some Common Problems in Log-Linear Analysis*. Sociological Methods and Research, 8-44.

See Also

loglm, measure_risk

Examples

```
## data.frame method
data(testdata2)
form <- ~ sex + water + roof
w <- "sampling_weight"

(modRisk(testdata2, method = "default", formulaM = form, weights = w))
(modRisk(testdata2, method = "CE", formulaM = form, weights = w))
(modRisk(testdata2, method = "PML", formulaM = form, weights = w))
(modRisk(testdata2, method = "weightedLLM", formulaM = form, weights = w))
(modRisk(testdata2, method = "IPF", formulaM = form, weights = w))

## application to a sdcMicroObj
data(testdata2)
sdc <- createSdcObj(testdata2,
```

```
  keyVars = c("urbrur", "roof", "walls", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight"
)
sdc <- modRisk(sdc, form = ~ sex + water + roof)
slot(sdc, "risk")$model



# an example using data from the laeken-pkg
library(laeken)
data(eusilc)
f <- as.formula(paste(" ~ ", "db040 + hsize + rb090 +
             age + pb220a + age:rb090 + age:hsize +
             hsize:rb090"))
w <- "rb050"
(modRisk(eusilc, method = "default", weights = w, formulaM = f, bound = 5))
(modRisk(eusilc, method = "CE", weights = w, formulaM = f, bound = 5))
(modRisk(eusilc, method = "PML", weights = w, formulaM = f, bound = 5))
(modRisk(eusilc, method = "weightedLLM", weights = w, formulaM = f, bound = 5))
```

`mvTopCoding` *Detection and winsorization of multivariate outliers*

Description

Imputation and detection of outliers

Usage

```
mvTopCoding(x, maha = NULL, center = NULL, cov = NULL, alpha = 0.025)
```

Arguments

`x`	an object coercible to a `data.table` containing numeric entries
`maha`	squared mahalanobis distance of each observation
`center`	center of data, needed for calculation of mahalanobis distance (if not provided)
`cov`	covariance matrix of data, needed for calcualtion of mahalanobis distance (if not provided)
`alpha`	significance level, determining the ellipsoide to which outliers should be placed upon

Details

Winsorizes the potential outliers on the ellipsoid defined by (robust) Mahalanobis distances in direction to the center of the data

Value

the imputed winsorized data

Author(s)

Johannes Gussenbauer, Matthias Templ

Examples

```
set.seed(123)
x <- MASS::mvrnorm(20, mu = c(5,5), Sigma = matrix(c(1,0.9,0.9,1), ncol = 2))
x[1, 1] <- 3
x[1, 2] <- 6
plot(x)
ximp <- mvTopCoding(x)
points(ximp, col = "blue", pch = 4)

# more dimensions
Sigma <- diag(5)
Sigma[upper.tri(Sigma)] <- 0.9
Sigma[lower.tri(Sigma)] <- 0.9
x <- MASS::mvrnorm(20, mu = rep(5,5), Sigma = Sigma)
x[1, 1] <- 3
x[1, 2] <- 6
pairs(x)

ximp <- mvTopCoding(x)
xnew <- data.frame(rbind(x, ximp))
xnew$beforeafter <- rep(c(0,1), each = nrow(x))
pairs(xnew, col = xnew$beforeafter, pch = 4)

# by hand (non-robust)
x[2,2] <- NA
m <- colMeans(x, na.rm = TRUE)
s <- cov(x, use = "complete.obs")
md <- stats::mahalanobis(x, m, s)
ximp <- mvTopCoding(x, center = m, cov = s, maha = md)
plot(x)
points(ximp, col = "blue", pch = 4)
```

---

nextSdcObj *nextSdcObj*

Description

internal function used to provide the undo-functionality.

Usage

```
nextSdcObj(obj)
```

Arguments

obj	a sdcMicroObj-class object

Value

a modified sdcMicroObj-class object

---

orderData_cpp *Reorder data*

Description

Reorders the data according to a column in the data set.
NOTE: This is an internal function used for testing the C++-function `orderData` which is used inside the C++-function `recordSwap()` to speed up performance.

Usage

```
orderData_cpp(data, orderIndex)
```

Arguments

data	micro data set containing only numeric values.
orderIndex	column index in `data` refering to the column by which data should be ordered.

Value

ordered data set.

---

plot.localSuppression *Plots for localSuppression objects*

Description

This function creates barplots to display the number of suppressed values in categorical key variables to achieve k-anonymity.

Usage

```
## S3 method for class 'localSuppression'
plot(x, ...)
```

Arguments

x	object of derived from localSuppression()
...	Additional arguments, currently available are: • "showDetails": logical, if set, a plot of suppressions by strata is shown (if possible)

Value

a ggplot plot object

Author(s)

Bernhard Meindl, Matthias Templ

See Also

localSuppression()

Examples

```
data(francdat)
```

---

plot.sdcMicroObj *Plotfunctions for objects of class sdcMicroObj*

Description

Descriptive plot function for sdcMicroObj-objects. Currently only visualization of local supression is implemented.

Usage

```
## S3 method for class 'sdcMicroObj'
plot(x, type = "ls", ...)
```

Arguments

x	An object of class sdcMicroObj
type	specified what kind of plot will be generated • "ls": plot of local suppressions in key variables
...	currently ignored

Value

a ggplot plot object or (invisible) NULL if local suppression using kAnon() has not been applied

Author(s)

Bernhard Meindl

Examples

```
data(testdata)
sdc <- createSdcObj(testdata,
  keyVars = c("urbrur", "roof", "walls", "relat", "sex"),
  w = "sampling_weight")
sdc <- kAnon(sdc, k = 3)
plot(sdc, type = "ls")
```

---

plotMicro *Comparison plots*

Description

Plots for the comparison of the original data and perturbed data.

Usage

```
plotMicro(x, p, which.plot = 1:3)
```

Arguments

x	an output object of `microaggregation()`
p	necessary parameter for the box cox transformation (`lambda`)
which.plot	which plot should be created? • 1: density traces • 2: parallel boxplots • 3: differences in totals

Details

Univariate and multivariate comparison plots are implemented to detect differences between the perturbed and the original data, but also to compare perturbed data which are produced by different methods.

Value

returns `NULL`; the selected plot is displayed

Author(s)

Matthias Templ

References

Templ, M. and Meindl, B., *Software Development for SDC in R*, Lecture Notes in Computer Science, Privacy in Statistical Databases, vol. 4302, pp. 347-359, 2006.

See Also

`microaggregation()`

Examples

```
data(free1)
df <- as.data.frame(free1)[, 31:34]
m1 <- microaggregation(df, method = "onedims", aggr = 3)
plotMicro(m1, p = 1, which.plot = 1)
plotMicro(m1, p = 1, which.plot = 2)
plotMicro(m1, p = 1, which.plot = 3)
```

---

pram *Post Randomization*

Description

To be used on categorical data stored as factors. The algorithm randomly changes the values of variables in selected records (usually the risky ones) according to an invariant probability transition matrix or a custom-defined transition matrix.

Usage

```
pram(obj, variables = NULL, strata_variables = NULL, pd = 0.8, alpha = 0.5)
```

Arguments

`obj`	Input data. Allowed input data are objects of class `data.frame`, `factor` or sdcMicroObj.
`variables`	Names of variables in `obj` on which post-randomization should be applied. If `obj` is a factor, this argument is ignored. Please note that pram can only be applied to factor-variables.
`strata_variables`	names of variables for stratification (will be set automatically for an object of class sdcMicroObj. One can also specify an integer vector or factor that specifies that desired groups. This vector must match the dimension of the input data set, however. For a possible use case, have a look at the examples.
`pd`	minimum diagonal entries for the generated transition matrix P. Either a vector of length 1 (which is recycled) or a vector of the same length as the number of variables that should be postrandomized. It is also possible to set `pd` to a numeric matrix. This matrix will be used directly as the transition matrix. The matrix must be constructed as follows:

- the matrix must be a square matrix
- the rownames and colnames of the matrix must match the levels (in the same order) of the factor-variable that should be postrandomized.
- the rowSums of the matrix need to equal 1

It is also possible to combine the different ways. For details have a look at the examples.

`alpha` amount of perturbation for the invariant Pram method. This is a numeric vector of length 1 (that will be recycled if necessary) or a vector of the same length as the number of variables. If one specified as transition matrix directly, `alpha` is ignored.

Value

a modified sdcMicroObj object or a new object containing original and post-randomized variables (with suffix "_pram").

Note

Deprecated method 'pram_strata' is no longer available in sdcMicro > 4.5.0

Author(s)

Alexander Kowarik, Matthias Templ, Bernhard Meindl

Kowarik, A. and Templ, M. and Meindl, B. and Fonteneau, F. and Prantner, B.: *Testing of IHSN Cpp Code and Inclusion of New Methods into sdcMicro*, in: Lecture Notes in Computer Science, J. Domingo-Ferrer, I. Tinnirello (editors.); Springer, Berlin, 2012, ISBN: 978-3-642-33626-3, pp. 63-77. doi:10.1007/9783642336270_6

Templ, M. and Kowarik, A. and Meindl, B.: *Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro.* in: Journal of Statistical Software, **67** (4), 1–36, 2015. doi:10.18637/jss.v067.i04

Templ, M.: *Statistical Disclosure Control for Microdata: Methods and Applications in R.* in: Springer International Publishing, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/9783-319502724

Examples

```
data(testdata)

## donttest is necessary because of
## Examples with CPU time > 2.5 times elapsed time
## caused by using C++ code and/or data.table
## using a factor variable as input
res <- pram(as.factor(testdata$roof))
print(res)
summary(res)

## using a data.frame as input
## pram can only be applied to factors
## -- > we have to recode to factors beforehand
```

```
testdata$roof <- factor(testdata$roof)
testdata$walls <- factor(testdata$walls)
testdata$water <- factor(testdata$water)

## pram() is applied within subgroups defined by
## variables "urbrur" and "sex"
res <- pram(
  obj = testdata,
  variables = "roof",
 strata_variables = c("urbrur", "sex"))
print(res)
summary(res)

## default parameters (pd = 0.8 and alpha = 0.5) for the generation
## of the invariant transition matrix will be used for all variables
res1 <- pram(
  obj = testdata,
  variables = c("roof", "walls", "water"))
print(res1)

## specific parameter settings for each variable
res2 <- pram(
   obj = testdata,
   variables = c("roof", "walls", "water"),
   pd = c(0.95, 0.8, 0.9),
   alpha = 0.5)
print(res2)

## detailed information on pram-parameters (such as the transition matrix 'Rs')
## is stored in the output, eg. for variable 'roof'
#attr(res2, "pram_params")$roof
## we can also specify a custom transition-matrix directly
mat <- diag(length(levels(testdata$roof)))
rownames(mat) <- colnames(mat) <- levels(testdata$roof)
res3 <- pram(
   obj = testdata,
   variables = "roof",
  pd = mat)
print(res3) # of course, nothing has changed!
## it is possible use a transition matrix for a variable and use the 'traditional' way
## of specifying a number for the minimal diagonal entries of the transision matrix
## for other variables. In this case we must supply `pd` as list.
res4 <- pram(
  obj = testdata,
  variables = c("roof", "walls"),
  pd = list(mat, 0.5),
  alpha = c(NA, 0.5))
print(res4)
summary(res4)
attr(res4, "pram_params")

## application to objects of class sdcMicro with default parameters
data(testdata2)
```

```
testdata2$urbrur <- factor(testdata2$urbrur)
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = c("roof", "walls", "water", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight")
sdc <- pram(
  obj = sdc,
  variables = "urbrur")
print(sdc, type = "pram")

## this is equal to the previous application. If argument 'variables' is NULL,
## all variables from slot 'pramVars' will be used if possible.
sdc <- createSdcObj(
   dat = testdata2,
  keyVars = c("roof", "walls", "water", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight",
  pramVars = "urbrur")
sdc <- pram(sdc)
print(sdc, type="pram")

## we can specify transition matrices for sdcMicroObj-objects too
testdata2$roof <- factor(testdata2$roof)
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = c("roof", "walls", "water", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight")
mat <- diag(length(levels(testdata2$roof)))

rownames(mat) <- colnames(mat) <- levels(testdata2$roof)
mat[1,] <- c(0.9, 0, 0, 0.05, 0.05)
sdc <- pram(
   obj = sdc,
   variables = "roof",
   pd = mat)
print(sdc, type = "pram")

## we can also have a look at the transitions
get.sdcMicroObj(sdc, "pram")$transitions
```

`print.freqCalc` *Print method for objects from class freqCalc.*

Description

Print method for objects from class freqCalc.

Usage

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

Arguments

x	object from class freqCalc
...	Additional arguments passed through.

Value

information about the frequency counts for key variables for object of class freqCalc.

Author(s)

Matthias Templ

See Also

freqCalc

Examples

```
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
```

---

print.indivRisk *Print method for objects from class indivRisk*

Description

Print method for objects from class indivRisk

Usage

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

Arguments

x	object from class indivRisk
...	Additional arguments passed through.

Value

few information about the method and the final correction factor for objects of class 'indivRisk'.

Author(s)

Matthias Templ

See Also

indivRisk

Examples

```
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f1 <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
data.frame(fk=f1$fk, Fk=f1$Fk)
## individual risk calculation:
indivRisk(f1)
```

---

print.localSuppression

Print method for objects from class localSuppression

Description

Print method for objects from class localSuppression

Usage

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

Arguments

x	object from class localSuppression
...	Additional arguments passed through.

Value

Information about the frequency counts for key variables for object of class 'localSuppression'.

Author(s)

Matthias Templ

See Also

localSuppression

Examples

```
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
l1 <- localSuppression(francdat, keyVars = c(2, 4, 5, 6))
l1
```

`print.micro` *Print method for objects from class micro*

Description

printing an object of class `micro`

Usage

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

Arguments

`x`	object from class micro
`...`	Additional arguments passed through.

Value

information about method and aggregation level from objects of class micro.

Author(s)

Matthias Templ

See Also

microaggregation

Examples

```
data(free1)
free1 <- as.data.frame(free1)
m1 <- microaggregation(free1[, 31:34], method='onedims', aggr=3)
m1
```

print.modrisk *Print method for objects from class modrisk*

Description

Print method for objects from class modrisk

Usage

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

Arguments

x	an object of class modrisk
...	Additional arguments passed through.

Value

Output of model-based risk estimation

Author(s)

Bernhard Meindl

See Also

modRisk

print.pram *Print method for objects from class pram*

Description

Print method for objects from class pram

Usage

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

Arguments

x	an object of class pram
...	Additional arguments passed through.

Value

absolute and relative frequencies of changed observations in each modified variable

Author(s)

Bernhard Meindl, Matthias Templ

Matthias Templ and Bernhard Meindl

See Also

pram

`print.sdcMicroObj` *Print and Extractor Functions for objects of class* `sdcMicroObj-class`

Description

Descriptive print function for Frequencies, local Supression, Recoding, categorical risk and numerical risk.

Usage

```
## S4 method for signature 'sdcMicroObj'
print(x, type = "kAnon", docat = TRUE, ...)
```

Arguments

`x`	An object of class `sdcMicroObj-class`
`type`	Selection of the content to be returned or printed
`docat`	logical, if TRUE (default) the results will be actually printed
`...`	the type argument for the print method, currently supported are:

- general: basic information on the input obj such as the number of observations and variables.
- kAnon: displays information about 2- and 3-anonymity
- ls: displays various information if local suppression has been applied.
- pram: displays various information if post-randomization has been applied.
- recode: shows information about categorical key variables before and after recoding
- risk: displays information on re-identification risks
- numrisk: displays risk- and utility measures for numerical key variables

Details

Possible values for the type argument of the print function are: "freq": for Frequencies, "ls": for Local Supression output, "pram": for results of post-randomization "recode":for Recodes, "risk": forCategorical risk and "numrisk": for Numerical risk.

Possible values for the type argument of the freq function are: "fk": Sample frequencies and "Fk": weighted frequencies.

Author(s)

Alexander Kowarik, Matthias Templ, Bernhard Meindl

Examples

```
data(testdata)

sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof','walls','relat','sex'),
  pramVars=c('water','electcon'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- microaggregation(sdc, method="mdav", aggr=3)
print(sdc)
print(sdc, type="general")
print(sdc, type="ls")
print(sdc, type="recode")
print(sdc, type="risk")
print(sdc, type="numrisk")
print(sdc, type="pram")
print(sdc, type="kAnon")
print(sdc, type="comp_numvars")
```

---

print.suda2 *Print method for objects from class suda2*

Description

Print method for objects from class suda2.

Usage

```
## S3 method for class 'suda2'
print(x, ...)
```

Arguments

x	an object of class suda2
...	additional arguments passed through.

Value

Table of dis suda scores.

Author(s)

Matthias Templ

See Also

suda2

`randSample_cpp` *Random Sampling*

Description

Randomly select records given a probability weight vector `prob`.
NOTE: This is an internal function used for testing the C++-function `randSample` which is used inside the C++-function `recordSwap()`.

Usage

```
randSample_cpp(ID, N, prob, IDused, seed)
```

Arguments

`ID`	vector containing record IDs from which to sample
`N`	integer defining the number of records to be sampled
`prob`	a vector of probability weights for obtaining the elements of the vector being sampled.
`IDused`	vector containing IDs which must not be sampled
`seed`	integer setting the sampling seed

rankSwap *Rank Swapping*

Description

Swapping values within a range so that, first, the correlation structure of original variables are preserved, and second, the values in each record are disturbed. To be used on numeric or ordinal variables where the rank can be determined and the correlation coefficient makes sense.

Usage

```
rankSwap(
  obj,
  variables = NULL,
  TopPercent = 5,
  BottomPercent = 5,
  K0 = NULL,
  R0 = NULL,
  P = NULL,
  missing = NA,
  seed = NULL
)
```

Arguments

obj	a sdcMicroObj-class-object or a data.frame
variables	names or index of variables for that rank swapping is applied. For an object of class sdcMicroObj-class, all numeric key variables are selected if variables=NULL.
TopPercent	Percentage of largest values that are grouped together before rank swapping is applied.
BottomPercent	Percentage of lowest values that are grouped together before rank swapping is applied.
K0	Subset-mean preservation factor. Preserves the means before and after rank swapping within a range based on K0. K0 is the subset-mean preservation factor such that $\|X_1 - X_2\| \leq \frac{2K_0X_1}{\sqrt{(N_S)}}$, where X_1 and X_2 are the subset means of the field before and after swapping, and N_S is the sample size of the subset.
R0	Multivariate preservation factor. Preserves the correlation between variables within a certain range based on the given constant R0. We can specify the preservation factor as $R_0 = \frac{R_1}{R_2}$ where R_1 is the correlation coefficient of the two fields after swapping, and R_2 is the correlation coefficient of the two fields before swapping.
P	Rank range as percentage of total sample size. We can specify the rank range itself directly, noted as P, which is the percentage of the records. So two records are eligible for swapping if their ranks, i and j respectively, satisfy $\|i-j\| \leq \frac{PN}{100}$, where N is the total sample size.

missing	missing - the value to be used as missing value in the C++ routine instead of NA. If NA, a suitable value is calculated internally. Note that in the returned dataset, all NA-values (if any) will be replaced with this value.
seed	Seed.

Details

Rank swapping sorts the values of one numeric variable by their numerical values (ranking). The restricted range is determined by the rank of two swapped values, which cannot differ, by definition, by more than P percent of the total number of observations. Only positive P, R0 and K0 are used and only one of it must be supplied. If none is supplied, sdcMicro sets parameter r0 to 0.95 internally.

Value

The rank-swapped data set or a modified sdcMicroObj-class object.

Author(s)

Alexander Kowarik for the interface, Bernhard Meindl for improvements.

For the underlying C++ code: This work is being supported by the International Household Survey Network (IHSN) and funded by a DGF Grant provided by the World Bank to the PARIS21 Secretariat at the Organisation for Economic Co-operation and Development (OECD). This work builds on previous work which is elsewhere acknowledged.

References

Moore, Jr.R. (1996) Controlled data-swapping techniques for masking public use microdata, U.S. Bureau of the Census *Statistical Research Division Report Series*, RR 96-04.

Kowarik, A. and Templ, M. and Meindl, B. and Fonteneau, F. and Prantner, B.: *Testing of IHSN Cpp Code and Inclusion of New Methods into sdcMicro*, in: Lecture Notes in Computer Science, J. Domingo-Ferrer, I. Tinnirello (editors.); Springer, Berlin, 2012, ISBN: 978-3-642-33626-3, pp. 63-77. doi:10.1007/9783642336270_6

Examples

```
data(testdata2)
data_swap <- rankSwap(
  obj = testdata2,
  variables = c("age", "income", "expend", "savings")
)

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = c("urbrur", "roof", "walls", "water", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight")
sdc <- rankSwap(sdc)
```

readMicrodata *readMicrodata*

Description

reads data from various formats into R. Used in sdcApp.

Usage

```
readMicrodata(
  path,
  type,
  convertCharToFac = TRUE,
  drop_all_missings = TRUE,
  ...
)
```

Arguments

path	a file path
type	which format does the file have. currently allowed values are • sas • spss • stata • R • rdf • csv
convertCharToFac	(logical) if TRUE, all character vectors are automatically converted to factors
drop_all_missings	(logical) if TRUE, all variables that contain NA-values only will be dropped
...	additional parameters. Currently used only if type='csv' to pass arguments to read.table().

Value

a data.frame or an object of class 'simple.error'. If a stata file was read in, the resulting data.frame has an additional attribute lab in which variable and value labels are stored.

Note

if type is either 'sas', 'spss' or 'stata', values read in as NaN will be converted to NA.

Author(s)

Bernhard Meindl

recordSwap *Targeted Record Swapping*

Description

Applies targeted record swapping on micro data considering the identification risk of each record as well the geographic topology.

Usage

```
recordSwap(data, ...)

## S3 method for class 'sdcMicroObj'
recordSwap(data, ...)

## Default S3 method:
recordSwap(
  data,
  hid,
  hierarchy,
  similar,
  swaprate = 0.05,
  risk = NULL,
  risk_threshold = 0,
  k_anonymity = 3,
  risk_variables = NULL,
  carry_along = NULL,
  return_swapped_id = FALSE,
  log_file_name = "TRS_logfile.txt",
  seed = NULL,
  ...
)
```

Arguments

data	must be either a micro data set in the form of a ‘data.table‘ or ‘data.frame‘, or an ‘sdcObject‘, see createSdcObj.
...	parameters passed to ‘recordSwap.default()‘
hid	column index or column name in ‘data‘ which refers to the household identifier.
hierarchy	column indices or column names of variables in ‘data‘ which refer to the geographic hierarchy in the micro data set. For instance county > municipality > district.
similar	vector or list of integer vectors or column names containing similarity profiles, see details for more explanations.
swaprate	double between 0 and 1 defining the proportion of households which should be swapped, see details for more explanations

`risk` either column indices or column names in ‘data‘ or ‘data.table‘, ‘data.frame‘ or ‘matrix‘ indicating risk of each record at each hierarchy level. If ‘risk‘-matrix is supplied to swapping procedure will not use the k-anonymity rule but the values found in this matrix for swapping. When using the risk parameter is expected to have assigned a maximum value in a household for each member of the household. If this condition is not satisfied, the risk parameter is automatically adjusted to comply with this condition. If risk parameter is provided then k-anonymity rule is suppressed.

`risk_threshold` single numeric value indicating when a household is considered "high risk", e.g. when this household must be swapped. Is only used when ‘risk‘ is not ‘NULL‘. Risk threshold indicates households that have to be swapped, but be aware that households with risk lower than threshold, but with still high enough risk may be swapped as well. Only households with risk set to 0 are not swapped. Risk and risk threshold must be equal or bigger then 0.

`k_anonymity` integer defining the threshold of high risk households (counts<k) for using k-anonymity rule

`risk_variables` column indices or column names of variables in ‘data‘ which will be considered for estimating the risk. Only used when k-anonymity rule is applied.

`carry_along` integer vector indicating additional variables to swap besides to hierarchy variables. These variables do not interfere with the procedure of finding a record to swap with or calculating risk. This parameter is only used at the end of the procedure when swapping the hierarchies. We note that the variables to be used as ‘carry_along‘ should be at household level. In case it is detected that they are at individual level (different values within ‘hid‘), a warning is given.

`return_swapped_id` boolean if ‘TRUE‘ the output includes an additional column showing the ‘hid‘ with which a record was swapped with. The new column will have the name ‘paste0(hid,"_swapped")‘.

`log_file_name` character, path for writing a log file. The log file contains a list of household IDs (‘hid‘) which could not have been swapped and is only created if any such households exist.

`seed` integer defining the seed for the random number generator, for reproducibility. if ‘NULL‘ a random seed will be set using ‘sample(1e5,1)‘.

Details

The procedure accepts a ‘data.frame‘ or ‘data.table‘ containing all necessary information for the record swapping, e.g parameter ‘hid‘, ‘similar‘, ‘hierarchy‘, etc ... First, the micro data in ‘data‘ is ordered by ‘hid‘ and the identification risk is calculated for each record in each hierarchy level. As of right now only counts is used as identification risk and the inverse of counts is used as sampling probability. NOTE: It will be possible to supply an identification risk for each record and hierarchy level which will be passed down to the C++-function. This is however not fully implemented.

With the parameter ‘k_anonymity‘ a k-anonymity rule is applied to define risky households in each hierarchy level. A household is set to risky if counts < k_anonymity in any hierarchy level and the household needs to be swapped across this hierarchy level. For instance, having a geographic hierarchy of NUTS1 > NUTS2 > NUTS3 the counts are calculated for each geographic variable and

defined ‘risk_variables‘. If the counts for a record falls below ‘k_anonymity‘ for hierarchy county (NUTS1, NUTS2, ...) then this record needs to be swapped across counties. Setting ‘k_anonymity = 0‘ disables this feature and no risky households are defined.

After that the targeted record swapping is applied, starting from the highest to the lowest hierarchy level and cycling through all possible geographic areas at each hierarchy level, e.g every county, every municipality in every county, etc, ...

At each geographic area, a set of values is created for records to be swapped. In all but the lowest hierarchy level, this is ONLY made out of all records which do not fulfil the k-anonymity and have not already been swapped. Those records are swapped with records not belonging to the same geographic area, which have not already been swapped beforehand. Swapping refers to the interchange of geographic variables defined in ‘hierarchy‘. When a record is swapped all other records containing the same ‘hid‘ are swapped as well.

At the lowest hierarchy level in every geographic area, the set of records to be swapped is made up of all records which do not fulfil the k-anonymity as well as the remaining number of records such that the proportion of swapped records of the geographic area is in coherence with the ‘swaprate‘. If due to the k-anonymity condition, more records have already been swapped in this geographic area then only the records which do not fulfil the k-anonymity are swapped.

Using the parameter ‘similar‘ one can define similarity profiles. ‘similar‘ needs to be a list of vectors with each list entry containing column indices of ‘data‘. These entries are used when searching for donor households, meaning that for a specific record the set of all donor records is made out of records which have the same values in ‘similar[[1]]‘. It is however important to note, that these variables can only be variables related to households (not persons!). If no suitable donor can be found the next similarity profile is used, ‘similar[[2]]‘ and the set of all donors is then made up out of all records which have the same values in the column indices in ‘similar[[2]]‘. This procedure continues until a donor record was found or all the similarity profiles have been used.

‘swaprate‘ sets the swaprate of households to be swapped, where a single swap counts for swapping 2 households, the sampled household and the corresponding donor. Prior to the procedure, the swaprate is applied on the lowest hierarchy level, to determine the target number of swapped households in each of the lowest hierarchies. If the target numbers of a decimal point they will randomly be rounded up or down such that the number of households swapped in total is in coherence to the swaprate.

Value

‘data.table‘ with swapped records.

Author(s)

Johannes Gussenbauer

Examples

```
# generate 10000 dummy households
library(data.table)
seed <- 2021
set.seed(seed)
nhid <- 10000
```

```
dat <- sdcMicro::createDat(nhid)

# define paramters for swapping
k_anonymity <- 1
swaprate <- .05 # 5%
similar <- list(c("hsize"))
hier <- c("nuts1", "nuts2")
risk_variables <- c("ageGroup", "national")
hid <- "hid"

## apply record swapping
#dat_s <- recordSwap(
#  data = dat,
#  hid = hid,
#  hierarchy = hier,
#  similar = similar,
#  swaprate = swaprate,
#  k_anonymity = k_anonymity,
#  risk_variables = risk_variables,
#  carry_along = NULL,
#  return_swapped_id = TRUE,
#  seed = seed
#)
#
## number of swapped households
#dat_s[hid != hid_swapped, uniqueN(hid)]
#
## hierarchies are not consistently swapped
#dat_s[hid != hid_swapped, .(nuts1, nuts2, nuts3, lau2)]
#
## use parameter carry_along
#dat_s <- recordSwap(
#   data = dat,
#   hid = hid,
#  hierarchy = hier,
#  similar = similar,
#  swaprate = swaprate,
#  k_anonymity = k_anonymity,
#  risk_variables = risk_variables,
#  carry_along = c("nuts3", "lau2"),
#  return_swapped_id = TRUE,
#  seed = seed)
#
#dat_s[hid != hid_swapped, .(nuts1, nuts2, nuts3, lau2)]
```

---

recordSwap_cpp *Targeted Record Swapping*

Description

Applies targeted record swapping on micro data set, see `?recordSwap` for details.
NOTE: This is an internal function called by the R-function `recordSwap()`. It's only purpose is to include the C++-function recordSwap() using Rcpp.

Usage

```
recordSwap_cpp(
  data,
  hid,
  hierarchy,
  similar_cpp,
  swaprate,
  risk,
  risk_threshold,
  k_anonymity,
  risk_variables,
  carry_along,
  log_file_name,
  seed = 123456L
)
```

Arguments

`data`	micro data set containing only integer values. A data.frame or data.table from R needs to be transposed beforehand so that data.size() ~ number of records - data.[0].size ~ number of varaibles per record. **NOTE:** *data has to be ordered by hid beforehand.*
`hid`	column index in `data` which refers to the household identifier.
`hierarchy`	column indices of variables in `data` which refers to the geographic hierarchy in the micro data set. For instance county > municipality > district.
`similar_cpp`	List where each entry corresponds to column indices of variables in `data` which should be considered when swapping households.
`swaprate`	double between 0 and 1 defining the proportion of households which should be swapped, see details for more explanations
`risk`	vector of vectors containing risks of each individual in each hierarchy level.
`risk_threshold`	double indicating risk threshold above every household needs to be swapped.
`k_anonymity`	integer defining the threshold of high risk households (k-anonymity). This is used as k_anonymity <= counts.
`risk_variables`	column indices of variables in `data` which will be considered for estimating the risk.
`carry_along`	integer vector indicating additional variables to swap besides to hierarchy variables. These variables do not interfere with the procedure of finding a record to swap with or calculating risk. This parameter is only used at the end of the procedure when swapping the hierarchies.

log_file_name	character, path for writing a log file. The log file contains a list of household IDs ('hid') which could not have been swapped and is only created if any such households exist.
seed	integer defining the seed for the random number generator, for reproducibility.

Value

Returns data set with swapped records.

`removeDirectID` *Remove certain variables from the data set inside a sdc object.*

Description

Delete variables without changing anything else in the sdcObject (writing NAs).

Usage

```
removeDirectID(obj, var)
```

Arguments

obj	object of class sdcMicroObj-class
var	name of the variable(s) to be remove

Value

the modified sdcMicroObj-class

Author(s)

Alexander Kowarik

Examples

```
## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata, keyVars=c('urbrur','roof'),
 numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- removeDirectID(sdc, var="age")
```

report *Generate an Html-report from an sdcMicroObj*

Description

Summary statistics of the original and the perturbed data set

Usage

```
report(
  obj,
  outdir = tempdir(),
  filename = "SDC-Report",
  title = "SDC-Report",
  internal = FALSE,
  verbose = FALSE
)
```

Arguments

obj	an object of class sdcMicroObj-class or reportObj
outdir	output folder
filename	output filename
title	Title for the report
internal	TRUE/FALSE, if TRUE a detailed internal report is produced, else a non-disclosive overview
verbose	TRUE/FALSE, if TRUE, some additional information is printed.

Details

The application of this function provides you with a html-report for your sdcMicro object that contains useful summaries about the anonymization process.

Author(s)

Matthias Templ, Bernhard Meindl

Examples

```
data(testdata2)
sdc <- createSdcObj(
  dat = testdata2,
  keyVars = c("urbrur", "roof", "walls", "water", "electcon", "relat", "sex"),
  numVars = c("expend", "income", "savings"),
  w = "sampling_weight"
)
report(sdc)
```

riskyCells *riskyCells*

Description

Allows to compute risky (unweighted) combinations of key variables either up to a specified dimension or using identification level. This mimics the approach taken in mu-argus.

Usage

```
riskyCells(obj, useIdentificationLevel = FALSE, threshold, ...)
```

Arguments

obj	a data.frame, data.table or an sdcMicroObj object
useIdentificationLevel	(logical) specifies if tabulation should be done up to a specific dimension (useIdentificationLevel = FALSE using argument maxDim) or taking identification levels (useIdentificationLevel = FALSE using argument level) into account.
threshold	a numeric vector specifiying the thresholds at which cells are considered to be unsafe. In case a tabulation is done up to a specific level (useIdentificationLevel = FALSE), the thresholds may be specified differently for each dimension. In the other case, the same threshold is used for all tables.
...	see possible arguments below • keyVars: index or variable-names within obj that should be used for tabulation. In case obj is a sdcMicroObj object, this argument is not used and the pre-defined key-variables are used. • level: in case useIdentificationLevel = TRUE, this numeric vector specifies the importance of the key variables. The construction of output tables follows the implementation in mu-argus, see e.g mu-argus. The length of this numeric vector must match the number of key variables. • maxDim: in case useIdentificationLevel = FALSE, this number specifies maximal number of variables to tablulate.

Value

a data.table showing the number of unsafe cells, thresholds for any combination of the key variables. If the input was a sdcMicroObj object and some modifications have been already applied to the categorical key variables, the resulting output contains the number of unsafe cells both for the original and the modified data.

Author(s)

Bernhard Meindl

Examples

```
## data.frame method / all combinations up to maxDim
# riskyCells(
#  obj = testdata2,
#  keyVars = 1:5,
#  threshold = c(50, 25, 10, 5),
#  useIdentificationLevel = FALSE,
#  maxDim = 4
# )
#riskyCells(
#  obj  = testdata2,
#  keyVars = 1:5,
#  threshold = 10,
#  useIdentificationLevel = FALSE,
#  maxDim = 3
#)
#
### data.frame method / using identification levels
#riskyCells(
#  obj = testdata2,
#  keyVars = 1:6,
#  threshold = 20,
#  useIdentificationLevel = TRUE,
#  level = c(1, 1, 2, 3, 3, 5)
#)
#riskyCells(
#  obj = testdata2,
#  keyVars = c(1, 3, 4, 6),
#  threshold = 10,
#  useIdentificationLevel = TRUE,
#  level = c(1, 2, 2, 4)
#)
#
### sdcMicroObj-method / all combinations up to maxDim
#testdata2[1:6] <- lapply(1:6, function(x) {
#  testdata2[[x]] <- as.factor(testdata2[[x]])
#})
#
#sdc <- createSdcObj(
#  dat = testdata2,
#  keyVars = c("urbrur", "roof", "walls", "water", "electcon", "relat", "sex"),
#  numVars = c("expend", "income", "savings"),
#  w = "sampling_weight")
#
#r0 <- riskyCells(
#  obj = sdc,
#  useIdentificationLevel=FALSE,
# threshold = c(20, 10, 5),
# maxDim = 3
#)
#
### in case key-variables have been modified, we get counts for
```

```
### original and modified data
#sdc <- groupAndRename(
#  obj = sdc,
#  var = "roof",
#  before = c("5", "6", "9"),
#  after = "5+"
#)
#r1 <- riskyCells(
#  obj = sdc,
#  useIdentificationLevel = FALSE,
#  threshold = c(10, 5, 3),
#  maxDim = 3
#)
#
### sdcMicroObj-method / using identification levels
#riskyCells(
#  obj = sdc,
#  useIdentificationLevel = TRUE,
#  threshold = 10,
#  level = c(1, 1, 3, 4, 5, 5, 5)
#)
```

`sampleDonor_cpp` *Random sample for donor records*

Description

Randomly select donor records given a probability weight vector. This sampling procedure is implemented differently than `randSample_cpp` to speed up performance of C++-function `recordSwap()`. **NOTE:** This is an internal function used for testing the C++-function `sampleDonor` which is used inside the C++-function `recordSwap()`.

Usage

```
sampleDonor_cpp(
  data,
  similar_cpp,
  hid,
  IDswap,
  IDswap_pool_vec,
  prob,
  seed = 123456L
)
```

Arguments

`data`	micro data containing the hierarchy levels and household ID
`similar_cpp`	List where each entry corresponds to column indices of variables in `data` which should be considered when swapping households.

hid	column index in data which refers to the household identifier.
IDswap	vector containing records for which a donor needs to be sampled
IDswap_pool_vec	set from which 'IDswap' was drawn
prob	a vector of probability weights for obtaining the elements of the vector being sampled.
seed	integer setting the sampling seed

---

sdcApp *sdcApp*

Description

starts the graphical user interface developed with *shiny*.

Usage

```
sdcApp(
  maxRequestSize = 50,
  debug = FALSE,
  theme = "IHSN",
  ...,
  shiny.server = FALSE
)
```

Arguments

maxRequestSize	(numeric) number defining the maximum allowed filesize (in megabytes) for uploaded files, defaults to 50MB
debug	logical if TRUE, set shiny-debugging options
theme	select stylesheet for the interface. Supported choices are • 'yeti' • 'flatly' • 'journal' • 'IHSN'
...	arguments (e.g host) that are passed through runApp when starting the shiny application
shiny.server	Setting this parameter to TRUE will return the app in the form of an object rather than invoking it. This is useful for deploying sdcApp via shiny-server.

Value

starts the interactive graphical user interface which may be used to perform the anonymization process.

Examples

```
if(interactive()) {
  sdcApp(theme = "flatly")
}
```

---

sdcMicroObj-class *Class* "sdcMicroObj"

Description

Class to save all information about the SDC process

Usage

```
createSdcObj(
  dat,
  keyVars,
  numVars = NULL,
  pramVars = NULL,
  ghostVars = NULL,
  weightVar = NULL,
  hhId = NULL,
  strataVar = NULL,
  sensibleVar = NULL,
  excludeVars = NULL,
  options = NULL,
  seed = NULL,
  randomizeRecords = FALSE,
  alpha = 1
)

undolast(object)

strataVar(object) <- value

## S4 replacement method for signature 'sdcMicroObj,characterOrNULL'
strataVar(object) <- value
```

Arguments

dat	The microdata set. A numeric matrix or data frame containing the data.
keyVars	Indices or names of categorical key variables. They must, of course, match with the columns of 'dat'.
numVars	Index or names of continuous key variables.
pramVars	Indices or names of categorical variables considered to be pramed.

ghostVars	if specified a list which each element being a list of exactly two elements. The first element must be a character vector specifying exactly one variable name that was also specified as a categorical key variable (keyVars), while the second element is a character vector of valid variable names (that must not be listed as keyVars). If localSuppression or kAnon was applied, the resulting suppression pattern for each key-variable is transferred to the depending variables.
weightVar	Indices or name determining the vector of sampling weights.
hhId	Index or name of the cluster ID (if available).
strataVar	Indices or names of stratification variables.
sensibleVar	Indices or names of sensible variables (for l-diversity)
excludeVars	which variables of dat should not be included in result-object? Users may specify a vector of variable-names available in dat that were not specified in either keyVars, numVars, pramVars, ghostVars, hhId, strataVar or sensibleVar.
options	additional options (if specified, a list must be used as input)
seed	(numeric) number specifiying the seed which will be set to allow for reproducablity. The number will be rounded and saved as element seed in slot options.
randomizeRecords	(logical) if TRUE, the order of observations in the input microdata set will be randomized.
alpha	numeric between 0 and 1 specifying the fraction on how much keys containing NAs should contribute to the frequency calculation which is also crucial for risk-estimation.
object	a sdcMicroObj-class object
value	NULL or a character vector of length 1 specifying a valid variable name

Value

a sdcMicroObj-class object

an object of class sdcMicroObj with modified slot @strataVar

Objects from the Class

Objects can be created by calls of the form new("sdcMicroObj", ...).

Author(s)

Bernhard Meindl, Alexander Kowarik, Matthias Templ, Elias Rut

References

Templ, M. and Meindl, B. and Kowarik, A.: *Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro*, Journal of Statistical Software, 67 (4), 1–36, 2015. doi:10.18637/jss.v067.i04

Examples

```
## we can also specify ghost (linked) variables
## these variables are linked to some categorical key variables
## and have the sampe suppression pattern as the variable that they
## are linked to after \code{\link{localSuppression}} has been applied
data(testdata)
testdata$electcon2 <- testdata$electcon
testdata$electcon3 <- testdata$electcon
testdata$water2 <- testdata$water

keyVars <- c("urbrur","roof","walls","water","electcon","relat","sex")
numVars <- c("expend","income","savings")
w <- "sampling_weight"

## we want to make sure that some variables not used as key-variables
## have the same suppression pattern as variables that have been
## selected as key variables. Thus, we are using 'ghost'-variables.
ghostVars <- list()

## we want variables 'electcon2' and 'electcon3' to be linked
## to key-variable 'electcon'
ghostVars[[1]] <- list()
ghostVars[[1]][[1]] <- "electcon"
ghostVars[[1]][[2]] <- c("electcon2","electcon3")


## donttest because Examples with CPU time > 2.5 times elapsed time
## we want variable 'water2' to be linked to key-variable 'water'
ghostVars[[2]] <- list()
ghostVars[[2]][[1]] <- "water"
ghostVars[[2]][[2]] <- "water2"

## create the sdcMicroObj
obj <- createSdcObj(testdata, keyVars=keyVars,
  numVars=numVars, w=w, ghostVars=ghostVars)

## apply 3-anonymity to selected key variables
obj <- kAnon(obj, k=3); obj

## check, if the suppression patterns are identical
manipGhostVars <- get.sdcMicroObj(obj, "manipGhostVars")
manipKeyVars <- get.sdcMicroObj(obj, "manipKeyVars")
all(is.na(manipKeyVars$electcon) == is.na(manipGhostVars$electcon2))
all(is.na(manipKeyVars$electcon) == is.na(manipGhostVars$electcon3))
all(is.na(manipKeyVars$water) == is.na(manipGhostVars$water2))

## exclude some variables
obj <- createSdcObj(testdata, keyVars=c("urbrur","roof","walls"), numVars="savings",
   weightVar=w, excludeVars=c("relat","electcon","hhcivil","ori_hid","expend"))
colnames(get.sdcMicroObj(obj, "origData"))
```

`selectHouseholdData` *Creates a household level file from a dataset with a household structure.*

Description

It removes individual level variables and selects one record per household based on a household ID. The function can also be used for other hierachical structures.

Usage

```
selectHouseholdData(dat, hhId, hhVars)
```

Arguments

`dat`	a data.frame with the full dataset
`hhId`	name of the variable with the household (cluster) ID
`hhVars`	character vector with names of all household level variables

Value

a data.frame with only household level variables and one record per household

Note

It is of great importance that users select a variable with containing information on household-ids and weights in `hhVars`.

Author(s)

Thijs Benschop and Bernhard Meindl

Examples

```
## ori-hid: household-ids; household_weights: sampling weights for households
x_hh <- selectHouseholdData(dat=testdata, hhId="ori_hid",
  hhVars=c("urbrur", "roof",  "walls", "water", "electcon", "household_weights"))
```

`set.sdcMicroObj` *set.sdcMicroObj*

Description

modify `sdcMicroObj-class`-objects depending on argument `type`

Usage

```
set.sdcMicroObj(object, type, input)
```

Arguments

`object`	a `sdcMicroObj-class`-object
`type`	a character vector of length 1 defining what to calculate|return|modify. Allowed types are listed below and the slot with the corresponding name will be replaced by the content of `input`.

- `origData`:
- `keyVars`:
- `pramVars`:
- `numVars`:
- `weightVar`:
- `hhId`:
- `strataVar`:
- `sensibleVar`:
- `manipPramVars`:
- `manipNumVars`:
- `manipGhostVars`:
- `manipStrataVar`:
- `risk`:
- `utility`:
- `pram`:
- `localSuppression`:
- `options`:
- `prev`:
- `set`:
- `additionalResults`:
- `deletedVars`:

`input`	a list depending on argument `type`. The content of the list must match the allowed data-type of the slot in the `sdcMicroObj-class`-object that should be replaced.

Value

a sdcMicroObj-class-object

Examples

```
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
ind_pram <- match(c("sex"), colnames(testdata2))
get.sdcMicroObj(sdc, type="pramVars")
sdc <- set.sdcMicroObj(sdc, type="pramVars", input=list(ind_pram))
get.sdcMicroObj(sdc, type="pramVars")
```

---

setLevels_cpp *Define Swap-Levels*

Description

Define hierarchy levels over which record needs to be swapped according to risk variables.
NOTE: This is an internal function used for testing the C++-function `setLevels()` which is applied inside `recordSwap()`.

Usage

```
setLevels_cpp(risk, risk_threshold)
```

Arguments

`risk`	vector of vectors containing risks of each individual in each hierarchy level. `risk[0]` returns the vector of risks for the first unit over all hierarchy levels. `risk[1]` the vector if risks for all hierarchy level of unit 2, and so on.
`risk_threshold`	double defining the risk threshold beyond which a record/household needs to be swapped. This is understood as risk>=risk_threshhold.

Value

Integer vector with hierarchy level over which record needs to be swapped with.

setRisk_cpp *Calculate Risk*

Description

Calculate risk for records to be swapped and donor records. Risks are defined by 1/counts, where counts is the number of records with the same values for specified `risk_variables` in the each geographic hierarchy. This risk will be used as sampling probability for both sampling set and donor set.
NOTE: This is an internal function used for testing the C++-function `setRisk` which is used inside the C++-function `recordSwap()`.

Usage

```
setRisk_cpp(data, hierarchy, risk_variables, hid)
```

Arguments

`data`	micro data set containing only numeric values.
`hierarchy`	column indices of variables in `data` which refere to the geographic hierarchy in the micro data set. For instance county > municipality > district.
`risk_variables`	column indices of variables in `data` which will be considered for estimating the risk.
`hid`	column index in `data` which refers to the household identifier.

show,sdcMicroObj-method *Show*

Description

show a sdcMicro object

Usage

```
## S4 method for signature 'sdcMicroObj'
show(object)
```

Arguments

`object`	an sdcmicro obj

Value

a sdcMicro object

Author(s)

Bernhard Meindl

`shuffle` *Shuffling and EGADP*

Description

Data shuffling and General Additive Data Perturbation.

Usage

```
shuffle(
  obj,
  form,
  method = "ds",
  weights = NULL,
  covmethod = "spearman",
  regmethod = "lm",
  gadp = TRUE
)
```

Arguments

`obj`	An object of class sdcMicroObj or a data.frame including the data.
`form`	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The responses have to consists of at least two variables of any class and the response variables have to be of class numeric. The response variables belongs to numeric key variables (quasi-identifiers of numeric scale). The predictors are can be distributed in any way (numeric, factor, ordered factor).
`method`	currently either the original form of data shuffling ("ds" - default), "mvn" or "mlm", see the details section. The last method is in experimental mode and almost untested.
`weights`	Survey sampling weights. Automatically chosen when obj is of class `sdcMicroObj-class`.
`covmethod`	Method for covariance estimation. "spearman", "pearson" and \ dQuotemcd are possible. For the latter one, the implementation in package robustbase is used.
`regmethod`	Method for multivariate regression. "lm" and "MM" are possible. For method "MM", the function "rlm" from package MASS is applied.
`gadp`	TRUE, if the egadp results from a fit on the original data is returned.

Details

Perturbed values for the sensitive variables are generated. The sensitive variables have to be stored as responses in the argument 'form', which is the usual formula interface for regression models in R.

For method "ds" the EGADP method is applied on the norm inverse percentiles. Shuffling then ranks the original values according to the GADP output. For further details, please see the references.

Method "mvn" uses a simplification and draws from the normal Copulas directly before these draws are shuffled.

Method "mlm" is also a simplification. A linear model is applied, the expected values are used as perturbed values before shuffling is applied.

Value

If 'obj' is of class `sdcMicroObj-class` the corresponding slots are filled, like manipNumVars, risk and utility. If 'obj' is of class "data.frame" an object of class "micro" with following entities is returned:

`shConf`	the shuffled numeric key variables
`egadp`	the perturbed (using gadp method) numeric key variables

Note

In this version, the covariance method chosen is used for any covariance and correlation estimations in the whole gadp and shuffling function.

Author(s)

Matthias Templ, Alexander Kowarik, Bernhard Meindl

References

K. Muralidhar, R. Parsa, R. Saranthy (1999). A general additive data perturbation method for database security. *Management Science*, 45, 1399-1415.

K. Muralidhar, R. Sarathy (2006). Data shuffling - a new masking approach for numerical data. *Management Science*, 52(5), 658-670, 2006.

M. Templ, B. Meindl. (2008). Robustification of Microdata Masking Methods and the Comparison with Existing Methods, in: *Lecture Notes on Computer Science*, J. Domingo-Ferrer, Y. Saygin (editors.); Springer, Berlin/Heidelberg, 2008, ISBN: 978-3-540-87470-6, pp. 14-25.

See Also

`rankSwap`, `lm`

Examples

```
data(Prestige,package="carData")
form <- formula(income + education ~ women + prestige + type, data=Prestige)
sh <- shuffle(obj=Prestige,form)
plot(Prestige[,c("income", "education")])
plot(sh$sh)
colMeans(Prestige[,c("income", "education")])
colMeans(sh$sh)
cor(Prestige[,c("income", "education")], method="spearman")
cor(sh$sh, method="spearman")

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- shuffle(sdc, method=c('ds'),regmethod= c('lm'), covmethod=c('spearman'),
form=savings+expend ~ urbrur+walls)
```

`subsetMicrodata` *subsetMicrodata*

Description

allows to restrict original data to only a subset. This may be useful to test some anonymization methods. This function will only be used in the graphical user interface `sdcApp`.

Usage

```
subsetMicrodata(obj, type, n)
```

Arguments

`obj`	an object of class `data.frame` containing micro data
`type`	algorithm used to sample from original microdata. Currently supported choices are
	`n_perc` the restricted microdata will be a `n-percent` sample of the original microdata.
	`first_n` only the first `n` observations will be used.
	`every_n` the restricted microdata set consists of every `n-th` record.
	`size_n` a total of `n` observations will be randomly drawn.
`n`	numeric vector of length 1 specifying the specific parameter with respect to argument `type`.

Value

an object of class `sdcMicroObj-class` with modified slot `@origData`.

Author(s)

Bernhard Meindl

`suda2` *Suda2: Detecting Special Uniques*

Description

SUDA risk measure for data from (stratified) simple random sampling.

Usage

```
suda2(obj, ...)
```

Arguments

`obj`	a `data.frame` or a sdcMicroObj-object
`...`	see arguments below

- `variables` Categorical (key) variables. Either the column names or and index of the variables to be used for risk measurement.
- `missing`: Missing value coding in the given data set.
- `DisFraction`: It is the sampling fraction for the simple random sampling, and the common sampling fraction for stratified sampling. By default, it's set to 0.01.
- `original_scores`: if this argument is `TRUE` (the default), the suda-scores are computed as described in paper "SUDA: A Program for Detecting Special Uniques" by Elliot et al., if `FALSE`, the computation of the scores is slightly different as it was done in the original implementation of the algorithm by the IHSN.

Details

Suda 2 is a recursive algorithm for finding Minimal Sample Uniques. The algorithm generates all possible variable subsets of defined categorical key variables and scans them for unique patterns in the subsets of variables. The lower the amount of variables needed to receive uniqueness, the higher the risk of the corresponding observation.

Value

A modified sdcMicroObj object or the following list

- `ContributionPercent`: The contribution of each key variable to the SUDA score, calculated for each row.
- `score`: The suda score 'disscore: The dis suda score
- `attribute_contributions:` a `data.frame` showing how much of the total risk is contributed by each variable. This information is stored in the following two variables:

  - `variable`: containing the name of the variable
  - `contribution`: contains how much risk a variable contributes to the total risk.
- `attribute_level_contributions`: returns risks of each attribute-level as a `data.frame` with the following three columns:
  - `variable`: the variable name
  - `attribute`: holding relevant level-codes
  - `contribution`: contains the risk of this level within the variable.

Note

Since version >5.0.2, the computation of suda-scores has changed and is now by default as described in the original paper by Elliot et al.

Author(s)

Alexander Kowarik and Bernhard Meindl (based on the C++ code from the Organisation For Economic Co-Operation And Development.

For the C++ code: This work is being supported by the International Household Survey Network and funded by a DGF Grant provided by the World Bank to the PARIS21 Secretariat at the Organisation for Economic Co-operation and Development (OECD). This work builds on previous work which is elsewhere acknowledged.

References

C. J. Skinner; M. J. Elliot (20xx) A Measure of Disclosure Risk for Microdata. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 64 (4), pp 855–867.

M. J. Elliot, A. Manning, K. Mayes, J. Gurd and M. Bane (20xx) SUDA: A Program for Detecting Special Uniques, Using DIS to Modify the Classification of Special Uniques

Anna M. Manning, David J. Haglin, John A. Keane (2008) A recursive search algorithm for statistical disclosure assessment. *Data Min Knowl Disc* 16:165 – 196

Templ, M. Statistical Disclosure Control for Microdata: Methods and Applications in R. *Springer International Publishing*, 287 pages, 2017. ISBN 978-3-319-50272-4. doi:10.1007/978331950272-4

`summary.freqCalc` *Summary method for objects from class freqCalc*

Description

Summary method for objects of class 'freqCalc' to provide information about local suppressions.

Usage

```
## S3 method for class 'freqCalc'
summary(object, ...)
```

Arguments

`object`	object from class freqCalc
`...`	Additional arguments passed through.

Details

Shows the amount of local suppressions on each variable in which local suppression was applied.

Value

Information about local suppression in each variable (only if a local suppression is already done).

Author(s)

Matthias Templ

See Also

`freqCalc`

Examples

```
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$fk
f$Fk
## individual risk calculation:
indivf <- indivRisk(f)
indivf$rk
## Local Suppression
localS <- localSupp(f, keyVar=2, threshold=0.25)
f2 <- freqCalc(localS$freqCalc, keyVars=c(4,5,6), w=8)
summary(f2)
```

`summary.micro` *Summary method for objects from class micro*

Description

Summary method for objects from class 'micro'.

Usage

```
## S3 method for class 'micro'
summary(object, ...)
```

Arguments

object	objects from class micro
...	Additional arguments passed through.

Details

This function computes several measures of information loss, such as

Value

meanx	A conventional summary of the original data
meanxm	A conventional summary of the microaggregated data
amean	average relative absolute deviation of means
amedian	average relative absolute deviation of medians
aonestep	average relative absolute deviation of onestep from median
devvar	average relative absolute deviation of variances
amad	average relative absolute deviation of the mad
acov	average relative absolute deviation of covariances
arcov	average relative absolute deviation of robust (with mcd) covariances
acor	average relative absolute deviation of correlations
arcor	average relative absolute deviation of robust (with mcd) correlations
acors	average relative absolute deviation of rank-correlations
adlm	average absolute deviation of lm regression coefficients (without intercept)
adlts	average absolute deviation of lts regression coefficients (without intercept)
apcaload	average absolute deviation of pca loadings
apppacaload	average absolute deviation of robust (with projection pursuit approach) pca loadings
atotals	average relative absolute deviation of totals
pmtotals	average relative deviation of totals

Author(s)

Matthias Templ

References

Templ, M. *Statistical Disclosure Control for Microdata Using the R-Package sdcMicro*, Transactions on Data Privacy, vol. 1, number 2, pp. 67-85, 2008. http://www.tdp.cat/issues/abs.a004a08.php

See Also

microaggregation, valTable

Examples

```
data(Tarragona)
m1 <- microaggregation(Tarragona, method = "onedims", aggr = 3)

summary(m1)
```

`summary.pram` *Summary method for objects from class pram*

Description

Summary method for objects from class 'pram' to provide information about transitions.

Usage

```
## S3 method for class 'pram'
summary(object, ...)
```

Arguments

`object`	object from class 'pram'
`...`	Additional arguments passed through.

Details

Shows various information about the transitions.

Value

The summary of object from class 'pram'.

Author(s)

Matthias Templ and Bernhard Meindl

References

Templ, M. *Statistical Disclosure Control for Microdata Using the R-Package sdcMicro*, Transactions on Data Privacy, vol. 1, number 2, pp. 67-85, 2008. http://www.tdp.cat/issues/abs.a004a08.php

See Also

`pram`

Examples

```
data(free1)
x <- as.factor(free1[,"MARSTAT"])
x2 <- pram(x)
x2
summary(x2)
```

Tarragona *Tarragona data set*

Description

A real data set comprising figures of 834 companies in the Tarragona area. Data correspond to year 1995.

Format

A data frame with 834 observations on the following 13 variables.

FIXED.ASSETS a numeric vector

CURRENT.ASSETS a numeric vector

TREASURY a numeric vector

UNCOMMITTED.FUNDS a numeric vector

PAID.UP.CAPITAL a numeric vector

SHORT.TERM.DEBT a numeric vector

SALES a numeric vector

LABOR.COSTS a numeric vector

DEPRECIATION a numeric vector

OPERATING.PROFIT a numeric vector

FINANCIAL.OUTCOME a numeric vector

GROSS.PROFIT a numeric vector

NET.PROFIT a numeric vector

Source

Public use data from the CASC project.

References

Brand, R. and Domingo-Ferrer, J. and Mateo-Sanz, J.M., Reference data sets to test and compare SDC methods for protection of numerical microdata. Unpublished. https://research.cbs.nl/casc/CASCrefmicrodata.pdf

Examples

```
data(Tarragona)
head(Tarragona)
dim(Tarragona)
```

`testdata` *A real-world data set on household income and expenditures*

Description

A concise (1-5 lines) description of the dataset.

Format

testdata: a data frame with 4580 observations on the following 15 variables.

urbrur a numeric vector

roof a numeric vector

walls a numeric vector

water a numeric vector

electcon a numeric vector

relat a numeric vector

sex a numeric vector

age a numeric vector

hhcivil a numeric vector

expend a numeric vector

income a numeric vector

savings a numeric vector

ori_hid a numeric vector

sampling_weight a numeric vector

household_weights a numeric vector

testdata2: A data frame with 93 observations on the following 19 variables.

urbrur a numeric vector

roof a numeric vector

walls a numeric vector

water a numeric vector

electcon a numeric vector

relat a numeric vector

sex a numeric vector

age a numeric vector

hhcivil a numeric vector

expend a numeric vector

income a numeric vector

savings a numeric vector

ori_hid a numeric vector

sampling_weight a numeric vector

represent a numeric vector

category_count a numeric vector

relat2 a numeric vector

water2 a numeric vector

water3 a numeric vector

References

The International Household Survey Network, www.ihsn.org

Examples

```
head(testdata)
head(testdata2)
```

`topBotCoding` *Top and Bottom Coding*

Description

Function for Top and Bottom Coding.

Usage

```
topBotCoding(obj, value, replacement, kind = "top", column = NULL)
```

Arguments

`obj`	a numeric vector, a `data.frame` or a `sdcMicroObj-class`-object
`value`	limit, from where it should be top- or bottom-coded
`replacement`	replacement value.
`kind`	top or bottom
`column`	variable name in case the input is a `data.frame` or an object of class `sdcMicroObj-class`.

Details

Extreme values larger or lower than `value` are replaced by a different value (`replacement` in order to reduce the disclosure risk.

Value

Top or bottom coded data or modified `sdcMicroObj-class`.

Note

top-/bottom coding of factors is no longer possible as of sdcMicro >=4.7.0

Author(s)

Matthias Templ and Bernhard Meindl

References

Templ, M. and Kowarik, A. and Meindl, B. Statistical Disclosure Control for Micro-Data Using the R Package sdcMicro. *Journal of Statistical Software*, **67** (4), 1–36, 2015. doi:10.18637/jss.v067.i04

See Also

`indivRisk`

Examples

```
data(free1)
res <- topBotCoding(free1[,"DEBTS"], value=9000, replacement=9100, kind="top")
max(res)

data(testdata)
range(testdata$age)
testdata <- topBotCoding(testdata, value=80, replacement=81, kind="top", column="age")
range(testdata$age)

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2, keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
          numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- topBotCoding(sdc, value=500000, replacement=1000, column="income")
testdataout <- extractManipData(sdc)
```

valTable *Comparison of different microaggregation methods*

Description

A Function for the comparison of different perturbation methods.

Usage

```
valTable(
  x,
 method = c("simple", "onedims", "clustpppca", "addNoise: additive", "swappNum"),
  measure = "mean",
  clustermethod = "clara",
  aggr = 3,
  nc = 8,
  transf = "log",
  p = 15,
  noise = 15,
  w = 1:dim(x)[2],
  delta = 0.1
)
```

Arguments

x	a data.frame or a matrix
method	character vector defining names of microaggregation-, adding-noise or rank swapping methods.
measure	FUN for aggregation. Possible values are mean (default), median, trim, onestep.
clustermethod	clustermethod, if a method will need a clustering procedure
aggr	aggregation level (default=3)
nc	number of clusters. Necessary, if a method will need a clustering procedure
transf	Transformation of variables before clustering.
p	Swapping range, if method swappNum has been chosen
noise	noise addition, if an addNoise method has been chosen
w	variables for swapping, if method swappNum has been chosen
delta	parameter for adding noise method "correlated2"

Details

Tabularize the output from summary.micro(). Will be enhanced to all perturbation methods in future versions.

Methods for adding noise should be named via addNoise:{method}, e.g. addNoise:correlated, where {method} specifies the desired method as described in addNoise().

Value

Measures of information loss splitted for the comparison of different methods.

Author(s)

Matthias Templ

References

Templ, M. and Meindl, B., Software Development for SDC in R, Lecture Notes in Computer Science, Privacy in Statistical Databases, vol. 4302, pp. 347-359, 2006.

See Also

microaggregation(), summary.micro()

Examples

```
data(Tarragona)

valTable(
  x = Tarragona[100:200, ],
  method=c("simple", "onedims", "pca")
)
```

---

varToFactor *Change the a keyVariable of an object of class sdcMicroObj-class from Numeric to Factor or from Factor to Numeric*

Description

Change the scale of a variable

Usage

```
varToFactor(obj, var)

varToNumeric(obj, var)
```

Arguments

obj	object of class sdcMicroObj-class
var	name of the keyVariable to change

Value

the modified sdcMicroObj-class

Examples

```
## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- varToFactor(sdc, var="urbrur")
```

`writeSafeFile` *writeSafeFile*

Description

writes an anonymized dataset to a file. This function should be used in the graphical user interface `sdcApp()` only.

Usage

```
writeSafeFile(obj, format, randomizeRecords, fileOut, ...)
```

Arguments

`obj`	a `data.frame` containing micro data
`format`	(character) specifies the output file format. Accepted values are: • `"rdata"`: output will be saved in the R binary file-format • `"sav"`: output will be saved as SPSS-file • `"dta"`: ouput will be saved as STATA-file • `"csv"`: output will be saved as comma seperated (text)-file • `"sas"`: output will be saved as SAS-file (sas7bdat)
`randomizeRecords`	(logical) specifies, if the output records should be randomized. The following options are possible: • `"no"`: default, no randomization takes place • `"simple"`: records are randomly swapped • `"byHH"`: if slot `"hhId"` is not `NULL`, the clusters defined by this variable are randomized across the dataset. If slot `"hhId"` is `NULL`, the records or the dataset are randomly changed. • `"withinHH"`: if slot `"hhId"` is not `NULL`, the clusters defined by this variable are randomized across the dataset and additionally, the order of records within the clusters are also randomly changed. If slot `"hhId"` is `NULL`, the records or the dataset are randomly changed.
`fileOut`	(character) file to which output should be written
`...`	optional arguments used for `utils::write.table()` if argument `"format"` equals `"csv"`

Value

invisible NULL if the file was successfully written

Author(s)

Bernhard Meindl

Index