Package ‘sdcMicro’

February 20, 2015

Type Package

Title Statistical Disclosure Control methods for anonymization of microdata and risk estimation

Version 4.4.0

Date 2014-07-18

Author Matthias Templ, Alexander Kowarik, Bernhard Meindl

Maintainer Matthias Templ <matthias.temp1@gmail.com>

Description Data from statistical agencies and other institutions are mostly confidential. This package can be used for the generation of anonymized (micro)data, i.e. for the creation of public- and scientific-use files. In addition, various risk estimation methods are included. Note that the package sdcMicroGUI includes a graphical user interface for various methods in this package.

LazyData TRUE

ByteCompile TRUE

LinkingTo Rcpp

Depends R (>= 2.10), brew, knitr.data.table,xtable

Suggests laeken

Imports car, robustbase, cluster, MASS, e1071, tools, Rcpp, methods, sets

License GPL-2

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

Collate '0classes.r' 'addNoise.r' 'aux_functions.r' 'dataGen.r'
'dRisk.R' 'dRiskRMD.R' 'dUtility.R' 'freqCalc.r'
'globalRecode.R' 'GUIfunctions.R' 'indivRisk.R'
'LLmodGlobalRisk.R' 'LocalRecProg.R' 'localSupp.R'
'localSupp2.R' 'localSupp2Wrapper.R' 'localSuppression.R'
'mdav.R' 'measure_risk.R' 'methods.r' 'microaggregation.R'
'plot.localSuppression.R' 'plotMicro.R' 'pram.R'
'print.freqCalc.R' 'print.indivRisk.R'
'print.localSuppression.R' 'print.micro.R' 'rankSwap.R'
R topics documented:

'shuffle.R' 'suda2.R' 'summary.freqCalc.R'
'summary.micro.R' 'summary.pram.R' 'swapNum.R'
'timeEstimation.R' 'topBotCoding.R' 'valTable.R' 'zzz.R'
'printFunctions.R' 'mafast.R' 'maG.R' 'show_sdcMicroObj.R'

NeedsCompilation: yes
Repository: CRAN
Date/Publication: 2014-07-18 15:20:55

R topics documented:

sdcmicro-package ........................................... 3
addNoise ...................................................... 8
calcRisks .................................................... 10
casc1 ......................................................... 11
CASCreffmicrodata ........................................... 12
dataGen ....................................................... 13
dRisk ........................................................ 14
dRiskRMD ...................................................... 15
dUtility ......................................................... 17
EIA ........................................................... 19
extractManipData .............................................. 22
francdat ....................................................... 23
free1 ........................................................ 24
freq-methods .................................................. 24
freqCalc ....................................................... 25
generateStrata ............................................... 27
globalRecode .................................................. 28
groupVars ...................................................... 29
indivRisk ...................................................... 30
LLmodGlobalRisk .............................................. 31
LocalRecProg .................................................. 33
localSupp ..................................................... 34
localSupp2 ..................................................... 36
localSupp2Wrapper .......................................... 38
localSuppression ............................................. 39
mafast ......................................................... 40
measure_risk .................................................. 42
microaggregation .............................................. 45
microaggrGower .............................................. 48
microData ..................................................... 50
plot.localSuppression ....................................... 50
plotMicro ..................................................... 51
pram ........................................................ 52
print.freqCalc ............................................... 53
print.indivRisk ............................................... 54
print.localSuppression ..................................... 55
print.micro ................................................... 56
Statistical Disclosure Control (SDC) for the generation of protected microdata for researchers and for public use.

Description

This package includes all methods of the popular software mu-Argus plus several new methods. In comparison with mu-Argus the advantages of this package are that the results are fully reproducible even with the included GUI, that the package can be used in batch-mode from other software, that the functions can be used in a very flexible way, that everybody could look at the source code and that there are no time-consuming meta-data management is necessary. However, the user should have a detailed knowledge about SDC when applying the methods on data.

The package is programmed using S4-classes and it comes with a well-defined class structure. The implemented graphical user interface (GUI) for microdata protection serves as an easy-to-handle tool for users who want to use the sdcMicro package for statistical disclosure control but are not used to the native R command line interface. In addition to that, interactions between objects which results from the anonymization process are provided within the GUI. This allows an automated recalculation and displaying information of the frequency counts, individual risk, information loss and data utility after each anonymization step. In addition to that, the code for every anonymization step carried out within the GUI is saved in a script which can then be easily modified and reloaded.
Details

Package: sdcMicro
Type: Package
Version: 2.5.9
Date: 2009-07-22
License: GPL 2.0

Author(s)

Matthias Templ, Alexander Kowarik, Bernhard Meindl
Maintainer: Matthias Templ <templ@statistik.tuwien.ac.at>

References


Examples

```r
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$f$k
f$f$K
## 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)
f2$f$K
## individual risk calculation:
```
indivf <- indivRisk(f)
indivf$rk
## Local Suppression
locals <- localSupp(f, keyVar=2, indivRisk=indivf$rk, threshold=0.25)
f2 <- freqCalc(locals$freqCalc, keyVars=c(2,4,5,6), w=8)
indivf2 <- indivRisk(f2)
indivf2$rk

## select another keyVar and run localSupp once again,
# if you think the table is not fully protected
data(free1)
f <- freqCalc(free1, keyVars=1:3, w=30)
ind <- indivRisk(f)
## and now you can use the interactive plot for individual risk objects:
## plot(ind)

## Local suppresion with localSupp2 and localSupp2Wrapper is more effective:
## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
l1 <- localSuppression(francdat, keyVars=c(2,4,5,6), importance=c(1,3,2,4))
l1
l1$x
l2 <- localSuppression(francdat, keyVars=c(2,4,5,6), k=2)
l3 <- localSuppression(francdat, keyVars=c(2,4,5,6), k=4)
## long computation time:
## l = localSupp2(free1, keyVar=1:3, w=30, k=2, importance=c(0.1,1,0.8))

## we want to avoid missings in column 5:
# l1 <- localSupp2Wrapper(francdat, keyVars=c(2,4,5,6), importance=c(1,1,0,1),
#   # w=8, kAnon=1)
# l1$x
## we want to avoid missings in column 5 and allow missings in 1 only if
## is really necessary:
# l1 <- localSupp2Wrapper(francdat, keyVars=c(2,4,5,6), importance=c(0.1,1,0,1),
#   # w=8, kAnon=1)
# l1$x
# plot(l1)

## Data from mu-Argus:
## Global recoding:
data(free1)
free1[, "AGE"] <- globalRecode(free1[, "AGE"], c(1,9,19,29,39,49,59,69,100), labels=1:8)

## Top coding:
topBotCoding(free1[, "DEBTS"], value=9000, replacement=9100, kind="top")

## Numerical Rank Swapping:
## do not use the mu-Argus test data set (free1)
# since the numerical variables are (probably) faked.
data(Tarragona)
Tarragona1 <- rankSwap(Tarragona, P=10)

## Microaggregation:
```r
m1 <- microaggregation(Tarragona, method="onedims", aggr=3)
m2 <- microaggregation(Tarragona, method="pca", aggr=3)
# summary(m1)
## approx. 1 minute computation time
## valTable(Tarragona, method=c("simple","onedims","pca"))

data(microData)
m1 <- microaggregation(microData, method="mdav")
x <- m1$x ### fix me
summary(m1)
plotMicro(m1, 0.1, which.plot=1) # too less observations...
data(free1)
plotMicro(microaggregation(free1[,31:34], method="onedims"), 0.1, which.plot=1)

## disclosure risk (interval) and data utility:
m1 <- microaggregation(Tarragona, method="onedims", aggr=3)
dRisk(obj=Tarragona, x=m1$x)
dRisk(obj=Tarragona, x=m2$x)
dUtility(obj=Tarragona, x=m1$x)
dUtility(obj=Tarragona, x=m2$x)

## S4 class code for Adding Noise methods will be included
# in the next version of sdcMicro.

## Fast generation of synthetic data with aprox.
# the same covariance matrix as the original one.
data(mtcars)
cov(mtcars[,4:6])
cov(dataGen(mtcars[,4:6],n=200))
pairs(mtcars[,4:6])
pairs(dataGen(mtcars[,4:6],n=200))

## PRAM
set.seed(123)
x <- factor(sample(1:4, 250, replace=TRUE))
pr1 <- pram(x)
length(which(pr1$xpramed == x))
x2 <- factor(sample(1:4, 250, replace=TRUE))
length(which(pram(x2)$xpramed == x2))
data(free1)
marstatPramed <- pram(free1,"MARSTAT")
## Not run:
## FOR OBJECTS OF CLASS sdcMicro
data(testdata)
sdc <- createSdcObj(testdata,
   keyVars=c("urbur","roof","walls","water","electcon","relat","sex"),
   numVars=c("expend","income","savings"), w="sampling_weight")
head(sdc@manipNumVars)
```
### Display Risks
sdc@risk$global
sdc <- dRisk(sdc)
sdc@risk$numeric
### use addNoise without Parameters
sdc <- addNoise(sdc, variables=c("expend","income"))
head(sdc@manipNumVars)
sdc@risk$numeric
### undolast
sdc <- undolast(sdc)
head(sdc@manipNumVars)
sdc@risk$numeric
### redo addNoise with Parameter
sdc <- addNoise(sdc, noise=0.2)
head(sdc@manipNumVars)
sdc@risk$numeric
### dataGen
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
sdc <- dataGen(sdc)
head(sdc@risk$individual)
sdc@risk$global
### LocalSuppression
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
sdc <- localSuppression(sdc)
head(sdc@risk$individual)
sdc@risk$global
### microaggregation
sdc <- undolast(sdc)
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc <- microaggregation(sdc)
head(get.sdcMicroObj(sdc, type="manipNumVars"))
### pram
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
sdc <- pram(sdc, keyVar="water")
head(sdc@risk$individual)
sdc@risk$global
### pram_strata
sdc <- undolast(sdc)
sdc <- pram_strata(sdc, variables=c("walls","water"))
head(sdc@risk$individual)
sdc@risk$global
### rankSwap
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc <- rankSwap(sdc)
addNoise

Adding noise to perturb data

Description

Various methods for adding noise to perturb continuous scaled variables.

Usage

addNoise(obj, variables, noise=150, method="additive", ...) #, p, delta)

Arguments

obj
variables
noise
method
... p
delta
either a data frame, matrix or a sdcMicroObj that should be perturbed
vector with names of variables that should be perturbed
amount of noise (in percentages)
dect’
see possible arguments below
multiplication factor for method ‘ROMM’
parameter for method ‘correlated2’, details can be found in the reference below.
Details

If `obj` is of class “sdcMicroObj” 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 there 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” 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

Methods

```r
signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")
```

Author(s)

Matthias Templ

References


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 an object of class 'sdcMicroObj'

... no arguments at the moment
Details

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

Methods

signature(obj = "sdcMicroObj")

See Also

sdcmicroobjMclass

testdata

data(testdata2)

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
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.

Usage
data(CASCrefmicrodata)

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFNLWGT</td>
<td>Final weight (2 implied decimal places)</td>
</tr>
<tr>
<td>AGI</td>
<td>Adjusted gross income</td>
</tr>
<tr>
<td>EMCONTRB</td>
<td>Employer contribution for hlth insurance</td>
</tr>
<tr>
<td>FEDTAX</td>
<td>Federal income tax liability</td>
</tr>
<tr>
<td>PTOTVAL</td>
<td>Total person income</td>
</tr>
<tr>
<td>STATETAX</td>
<td>State income tax liability</td>
</tr>
<tr>
<td>TAXINC</td>
<td>Taxable income amount</td>
</tr>
<tr>
<td>POTHVAL</td>
<td>Total other persons income</td>
</tr>
<tr>
<td>INTVAL</td>
<td>Amt of interest income</td>
</tr>
<tr>
<td>PEARNVAL</td>
<td>Total person earnings</td>
</tr>
<tr>
<td>FICA</td>
<td>Soc. sec. retirement payroll deduction</td>
</tr>
<tr>
<td>WSALVAL</td>
<td>Amount: Total Wage and salary</td>
</tr>
<tr>
<td>ERNVAL</td>
<td>Business or Farm net earnings</td>
</tr>
</tbody>
</table>

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

References

Examples
data(CASCrefmicrodata)
str(CASCrefmicrodata)
Fast generation of synthetic data

Description

Fast generation of (primitive) synthetic multivariate normal data.

Usage

dataGen(obj,...)# n = 200

Arguments

obj  data.frame or matrix
... see possible arguments below
n   amount of observations for the generated data

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.

Methods

signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcmicroObj")

Note

With this method only multivariate normal distributed data with approximately 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

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('urbur', 'roof', 'walls', 'water', 'electcon', 'relat', 'sex'),
  numVars=c('expend', 'income', 'savings'), w='sampling_weight')
sdc <- dataGen(sdc)

---

**dRisk**

*overal disclosure risk*

---

Description

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

Usage

dRisk(obj,...) # xm, k = 0.05

Arguments

- `obj`: original data or object of class sdcMicroObj
- `...`: see possible arguments below
- `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”
Methods

signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcmicroObj")

Author(s)

Matthias Templ

References


See Also

dUtility, dUtility

Examples

data(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('urbur', '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,...) #xm, k = 0.01, k2=0.05
Arguments

- **obj**: original data or object of class `sdcMicroObj`
- **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 observation. 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”

- **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
Methods

signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcmicroObj")

Author(s)

Matthias Templ

References


See Also
dRisk

Examples

data(Tarragona)
x <- Tarragona[, 5:7]
y <- addNoise(x)$x
RiskRMD(x, x=y)
Risk(x, x=y)

data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c('urbrur','roof','walls','water','electric','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
## this is already made internally:
## sdc <- dRiskRMD(sdc)
## and already stored in sdc

---

dUtility data utility

Description

IL1s data utility.

Usage

dUtility(obj,...)#, x, method="IL1")
Arguments

obj: original data or object of class sdcMicroObj

... see arguments below

xm: perturbed data

method: method IL1 or eigen. More methods are implemented in summary.micro()

Details

The standardised distances of the perturbed data values to the original ones are measured. Measure IL1 measures the distances between the original values and the perturbed ones, scaled by the standard deviation. Method ‘eigen’ and ‘robeigen’ compares the eigenvalues and robust eigenvalues form the original data and the perturbed data.

Value

data utility or modified entry for data utility the “sdcMicroObj”.

Methods

signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")

Author(s)

Matthias Templ

References

for IL1s: see http://vneumann.etse.urv.es/publications/sci/lncs3050Outlier.pdf,


See Also

dRisk, dRiskRMD

Examples

data(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, x=m=y)
dRisk(x, x=m=y)
dUtility(x, x=m=y)
dUtility(x, x=m=y, method="eigen")
dUtility(x, x=m=y, method="robeigen")

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

**EIA**

**EIA data set**

**Description**

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

**Usage**

```r
data(EIA)
```

**Format**

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

- **UTILITYID** UNIQUE UTILITY IDENTIFICATION NUMBER
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

Cuyivre 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

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

Greenview City of

Gulf Power Company

Gulf States Utilities Co

Hasting Utilities

Hawaii Electric Light Co Inc

Henderson-McCoy Utilities

Homer Electric Assn Inc

Hot Springs Rural El Assn

Inc Houston Lighting & Power Co

Huntsville City of

Idaho Power Co

IES Utilities Inc

Illinois City of

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 of

Jones-Onslow Elec Member Corp

Kansas City City of

Kaskaskia Power & Light Co

Kentucky Power 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 of

Lincoln City of

Los Angeles City of

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 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

Palmietto 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

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 Pro"j Ag I & P Dist

San Antonio City of

Savannah Electric & Power Co

Seattle City of

Sierra Pacific Power Co

Singing River Elec Power Assn
|------------------------|----------------|-----------------------------|-----------------------------|--------------------------|-----------------------------|-----|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|--------------------------|------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|-----------------------------|

**STATE**  STATE FOR WHICH THE UTILITY IS REPORTING. A factor with levels AK AL AR AZ CA CO CT 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 SC SD TN TX UT VA VT WA WI WY 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

**OTHSALES**  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**


**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'.

Usage

extractManipData(obj, ignoreKeyVars=FALSE, ignorePramVars=FALSE, ignoreNumVars=FALSE, ignoreStrataVar=FALSE)

Arguments

- **obj**: object of class 'sdcMicroObj'
- **ignoreKeyVars**: If manipulated KeyVariables should be returned or the unchanged original variable
- **ignorePramVars**: If manipulated PramVariables should be returned or the unchanged original variable
- **ignoreNumVars**: If manipulated NumericVariables should be returned or the unchanged original variable
- **ignoreStrataVar**: If manipulated StrataVariables should be returned or the unchanged original variable

Value

a data frame

Methods

signature(obj = "sdcMicroObj")

Author(s)

Alexander Kowarik

Examples

```r
# 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

Usage

data(francdat)

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


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.

Usage
data(free1)

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-methods

Print and Extractor Functions for objects of class ‘sdcMicroObj’

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

Usage
print(x,...)
freq(obj, type="fk")
freqCalc

Arguments

x  An object of class 'sdcMicroObj'
obj An object of class 'sdcMicroObj'
type Selection of the content to be returned or printed-
... the type argument for the print method

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":
for Categorical 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

Examples

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

Description

Computation and estimation of the sample and population frequency counts.

Usage

```r
freqCalc(x, keyVars, w = NULL, fast=TRUE)
```
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.
- **fast**: beta version of faster algorithm should not change the results in any way

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 (categorical) key variables, since the computation is done in C.

`freqCalc()` does not support sdcMicro S4 class objects.

Value

Object from class freqCalc.

- **freqCalc**: data
- **keyVars**: keyVars
- **w**: index of weight vector. NULL if you do not have a sample.
- **indexG**: 
- **fk**: the frequency of equal observations in the key variables subset sample given for each observation.
- **Fk**: estimated frequency in the population
- **n1**: amount of observations with \( fk=1 \)
- **n2**: amount of observations with \( fk=2 \)

Author(s)

Bernhard Meindl and Matthias Templ

References


generateStrata

See Also

\texttt{indivRisk, measure\_risk}

Examples

data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$f$freqCalc
f$f$k
f$f$K
## 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)
f2$f$K

# time comparison freqCalc old version vs. new version
data(testdata)
system.time( f3 <- freqCalc(testdata,keyVars=c(1:4,7),w=14,fast=FALSE) )
system.time( f3f <- freqCalc(testdata,keyVars=c(1:4,7),w=14,fast=TRUE) )

---

\texttt{generateStrata} \hspace{1cm} \textit{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

\texttt{generateStrata(df, stratavars, name)}

Arguments

df \hspace{1cm} a data.frame
stratavars \hspace{1cm} character vector with variable name
name \hspace{1cm} name of the newly generated variable

Value

The original data set with one new column.

Author(s)

Alexander Kowarik
Examples

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

---

**globalRecode**  
*Global Recoding*

**Description**

Global recoding

**Usage**

```r
globalRecode(obj,...)#, column,breaks, labels, method="equidistant")
```

**Arguments**

- **obj**: vector of class numeric or of class factor with integer labels for recoding or an object of class `sdcMicroObj`
- **column**: which keyVar should be changed
- **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**: method "equidistant" for equal sized intervalls
  - method "logEqui" for equal sized intervalls for log-transformed data
  - method "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” or a factor, unless labels = FALSE which results in the mere integer level codes.

**Methods**

```r
signature(obj = "ANY")
signature(obj = "sdcMicroObj")
```
groupVars

**See Also**

cut

**Examples**

data(free1)
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))
table(globalRecode(free1[, "AGE"], breaks=c(1, 9, 19, 29, 39, 49, 59, 69, 100)))
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="urbrur", breaks=5)

---

**groupVars**

Join levels of a keyVariable in an object of class 'sdcMicroObj'

**Description**

Transforms the factor variable into a factors with less levels and recomputes risk.

**Usage**

groupVars(obj, var, before, after)

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>object of class 'sdcMicroObj'</td>
</tr>
<tr>
<td>var</td>
<td>name of the keyVariable to change</td>
</tr>
<tr>
<td>before</td>
<td>vector of levels before</td>
</tr>
<tr>
<td>after</td>
<td>vector of levels after</td>
</tr>
</tbody>
</table>

**Value**

the modified “sdcMicroObj”

**Methods**

signature(obj = "sdcMicroObj")
Examples

```r
## 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 <- groupVars(sdc, var="urbrur", before=c(1,2), after=1)
```

### indivRisk

**Individual Risk computation**

Description

Individual risk computation.

Usage

```r
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 one have survey data and FALSE if one deal with the whole population.

Details

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.

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


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

---

LLmodGlobalRisk  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

LLmodGlobalRisk(obj, method = "IPF", inclProb = NULL, form = NULL, modOutput = FALSE)
**Arguments**

- **obj**: An object of class `sdcMicroObj` or a numeric matrix or data frame containing the categorical key variables.
- **method**: At this time, only iterative proportional fitting ("IPF") can be used.
- **incProb**: Inclusion probabilities (experimental)
- **form**: A formula specifying the model.
- **modOutput**: If TRUE, additional output is given.

**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 (SU) as

\[ \tau_1 = \sum_{SU} P(F_k = 1 | f_k = 1) \]

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

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

\[ \tau_2 = \sum_{SU} P(F_k = 1 | f_k = 1) \], CORRECT!

Since population frequencies \( F_k \) are unknown, they has 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 or the modified risk in the "sdcMicroObj" object.

**Author(s)**

Matthias Templ

**References**


**See Also**

`loglm`, `measure_risk`
Examples

```r
data(testdata2)
x <- testdata2[,c("sex","water","roof")]
res <- LLmodGlobalRisk(x, form=-sex+water+roof,
inclProb=1/testdata2[,"sampling_weight"])
res$gr1; res$gr2
```

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

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.

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-anonymity), one can form new categories based on the recoding result and repeatedly apply this algorithm.

Usage

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

Arguments

- `obj` Input data or object of class sdcMicroObj
- `ancestors` Names of ancestors of the categorical 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
**Value**

dataframe with original variables and the supressed variables (suffix _lr). / the modified “sdcMicroObj”

**Methods**

signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")

**Author(s)**

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

**References**


**Examples**

```r
# LocalRecProg
data(testdata2)
r1=LocalRecProg(testdata2,
categorical=c("urbrur", "roof", "walls", "water", "sex", "relat"),
missingValue=-99)
r2=LocalRecProg(testdata2,
categorical=c("urbrur", "roof", "walls", "water", "sex", "relat"),
ancestor=c("water2", "water3", "relat2"),
ancestor_setting=c("water","water","relat"),missingValue=-99)
r3=LocalRecProg(testdata2,
categorical=c("urbrur", "roof", "walls", "water", "sex", "relat"),
ancestor=c("water2", "water3", "relat2"),
ancestor_setting=c("water","water","relat"),missingValue=-99,
FindLowestK=FALSE)
```

```r
## 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 <- LocalRecProg(sdc)
```

---

**Local Suppression**

**Description**

A simple method to perform local suppression.
Usage

localSupp(obj, threshold=0.15, keyVar,...)# indivRisk)

Arguments

  obj  object of class freqCalc or sdcMicroObj
  threshold  threshold for individual risk
  keyVar  Variable on which some values might be suppressed
  ...  see arguments below
  indivRisk  object from class indivRisk

Details

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

Value

Manipulated data with suppressions or the “sdcMicroObj” object with manipulated data.

Methods

signature(obj = "ANY")

signature(obj = "sdcMicroObj")

Author(s)

Matthias Templ

References


See Also

freqCalc, indivRisk

Examples

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

```r
# Local Suppression
localS <- localSupp(f, keyVar=2, indivRisk=indivf$rk, threshold=0.25)
f2 <- freqCalc(localS$freqCalc, keyVars=c(4,5,6), w=8)
indivf2 <- indivRisk(f2)
```

## Description

An Algorithm to perform local suppression to achieve k-anonymity.

## Usage

```r
localSupp2(x, keyVars, w, importance=rep(1, length(keyVars)),
method="minimizeSupp", k=1)
```

## Arguments

- `x`: data frame or matrix
- `keyVars`: column index of key variables
- `w`: column index of sampling weights
- `importance`: weights for each key variable
- `method`: "minimizeSupp" (default), further methods will be included in future versions of the package
- `k`: parameter for k-anonymity.

## Details

With the help of this algorithm you can achieve k-anonymity in an optimized way. The procedure set missings only to those key variables for which the importance is greater than 0. Key variables with higher importance will be prefered to be the variable which will used for suppression of specific values, i.e. the vector of importance assign to each key variables a weight which is considered by the algorithm.

To guarantee k-anonymity the wrapper of function localSupp2 should be applied (localSupp2Wrapper())
However, if the importance of some key variables are equal to zero, the algorithm may not find a k-anonymity solution (because there isn’t any solution reachable at all, for example). The easiest way to overcome this situation is to re-run the algorithm and allow for NA’s in some more key variables, i.e. re-run the algorithm with importance greater than 0 for all entries of importance. This will result in k-anonymized results and leads to only few suppressions in the key variables where the importance of the variables are considered.

Method fastSupp avoids some calculation steps but this method is only significant faster if there is a large data sets with few key variables. However, fastSupp leads to an oversuppression (slightly).

Value

Object from class localSupp2.

- xAnon: resulting data with suppressions
- supps: number of suppressions in the key variables
- totalSupps: total number of suppressions.
- anonymity: TRUE, if k-anonymity is achieved
- keyVars: index of the key variables.
- importance: weight vector for key variables
- k: k for k-anonymity

Note

fix me: Implementation in C and interface to R.

Author(s)

Matthias Templ, Bernhard Meindl

References


See Also

freqCalc, localSuppression

Examples

print("this function is deprecated, please use localSuppression() instead")
**Description**

A wrapper function for function localSupp2 in order to guarantee k-anonymity.

**Usage**

```r
localSupp2Wrapper(x, keyVars, w, importance=rep(1, length(keyVars)),
method="minimizeSupp", kAnon=2)
```

**Arguments**

- `x`: data frame or matrix
- `keyVars`: column index of key variables
- `w`: column index of sampling weights
- `importance`: weights for each key variable, see ‘localSupp2()’
- `method`: “minimizeSupp” (default), further methods will be included in future versions of the package
- `kAnon`: parameter for k-anonymity.

**Details**

This wrapper function guarantees k-anonymity. If function localSupp2() cannot be reach k-anonymity, localSupp2 must be re-run on the previous results as long as k-anonymity is reached. If k-anonymity cannot be achieved (because the entries of parameter importance includes too much zeros) the function breaks after a sub-optimal solution is obtained.

**Value**

Object from class localSupp2.

```
xAnon: resulting data with suppressions
supps: number of suppressions in the key variables
totalSupps: total number of suppressions.
anonymity: TRUE, if k-anonymity is achieved
keyVars: index of the key variables.
importance: weight vector for key variables
kAnon: k for k-anonymity
```

**Note**

fix me: Implementation in C and interface to R.
Author(s)
Bernhard Meindl, Matthias Templ

References

See Also
freqCalc, localsuppression

Examples
print("this function is deprecated, please use localsuppression() instead")

localSuppression     Local Suppression to obtain k-anonymity

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

Usage
localSuppression(obj,k=2, importance=NULL,...)#, keyVars)

Arguments

obj an object of class sdcMicroObj or a data frame or matrix
k threshold for k-anonymity
importance numeric vector of numbers between 1 and n (n=length of vector keyVars). This vector represents the "importance" of variables that should be used for local suppression in order to obtain k-anonymity. key-variables with importance=1 will - if possible - not suppressed, key-variables with importance=n will be used whenever possible.
... see arguments below
keyVars numeric vector specifying indices of (categorical) key-variables

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.
Value
Manipulated data set with suppressions that has k-anonymity with respect to specified key-variables or the manipulated data stored in the “sdcMicroObj” object.

Methods

\[
\text{signature(obj = "data.frame")}
\]
\[
\text{signature(obj = "matrix")}
\]
\[
\text{signature(obj = "sdcMicroObj")}
\]

Author(s)

Bernhard Meindl, Matthias Templ

Examples

data(francdat)
## Local Suppression
locals5 <- localSuppression(francdat, keyVar=c(4,5,6))
locals5
plot(locals5)

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

mafast 

**Fast and Simple Microaggregation**

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

Usage

\[
\text{mafast(obj, variables=NULL, by=NULL, aggr=3, measure=mean)}
\]

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>either an object of class sdcMicroObj or a data frame or matrix</td>
</tr>
<tr>
<td>variables</td>
<td>variables to microaggregate. If obj is of class sdcMicroObj the numerical key variables are chosen per default.</td>
</tr>
<tr>
<td>by</td>
<td>grouping variable for microaggregation. If obj is of class sdcMicroObj the strata variables are chosen per default.</td>
</tr>
<tr>
<td>aggr</td>
<td>aggregation level (default=3)</td>
</tr>
<tr>
<td>measure</td>
<td>aggregation statistic, mean, median, trim, onestep (default = mean)</td>
</tr>
</tbody>
</table>
Value

If `obj` was of class “sdcMicroObj” 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.

Methods

signature(obj = "ANY")
signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")

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'),
umVars=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

## Not run:
#### Performance tests
x <- testdata
for(i in 1:20){
  x <- rbind(x,testdata)
}
system.time(xx <= mafast(x,variables=c("expend","income","savings"),aggr=50,by="sex"))

## End(Not run)
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.

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.

Usage

measure_risk(obj,...)
#measure_risk(data,keyVars,w=NULL,missing=-999,
#hid=NULL,max_global_risk=.01,fast_hier=TRUE)
ldiversity(obj,ldiv_index,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 “sdcMicroObjet”
... see arguments below
data Input data, either a matrix or a data.frame.
keyVars Names of categorical key variables
w name of variable containing sample weights
hid name of the clustering variable, e.g. the household ID
missing a integer value to be used as missing value in the C++ routine
ldiv_index indices (or names) of the variables used for l-diversity
l_recurs_c l-Diversity Constant
x Output of measure_risk, measure_hier or measure_thres
max_global_risk Maximal global risk for threshold computation
fast_hier If TRUE a fast approximation is computed if household data are provided.
Details

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-diversity 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 “sdeMicroObj” object or a list with the following elements:

- `global_risk_ER` expected number of re-identification.
- `global_risk` global risk (sum of individual 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 hierarchical 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 individual 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.
Methods

signature(obj = "data.frame") Method for object of class “data.frame”
signature(obj = "matrix") Method for object of class “matrix”
signature(obj = "sdcmicroobj") Method for object of S4 class “sdcmicroobj”

Author(s)

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

References

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

See Also

freqCalc, indivRisk

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:
resi <- measure_risk(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"))
print(resi)
head(res$i$Res)
reswi <- measure_risk(testdata,
  keyVars=c("urbrur","roof","walls","water","sex"),w="sampling_weight")
print(reswi)
head(resw$i$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")
## already interally applied and available in object sdc:
## sdc <- measure_risk(sdc)

### 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", opt = FALSE, measure = "mean",
  trim = 0, varsort = 1, transf = "log")

**Arguments**

- **obj**: either an object of class sdcMicroObj or a data frame or matrix
- **variables**: variables to microaggregate. For NULL: If obj is of class sdcMicroObj the categorical key variables are chosen per default. For data.frames and matrices all columns are chosen per default.
- **aggr**: aggregation level (default=3)
- **strata_variables**: by-variables for applying microaggregation only within strata defined by the variables
- **method**: pca, rmd, onedims, single, simple, clustpca, pppca, clustpppca, mdav, clustmdpca, influence, mcdpca
- **nc**: number of cluster, if the chosen method performs cluster analysis
- **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.
microaggregation

clustermethod  clustermethod, if necessary
opt            experimental
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 [http://neon.vb.cbs.nl/casc/Glossary.htm](http://neon.vb.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 measures 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 ‘oneds’, however, often this method is named ‘individual ranking’), but multivariate statistics are strongly 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” 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:
microaggregation

mx the aggregated data
x original data
method method
aggr aggregation level
measure proximity measure for aggregation
fot correction factor, necessary if totals calculated and n divided by aggr is not an integer.

Methods

signature(obj = "ANY")
signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")

Author(s)

Matthias Templ

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

http://www.springerlink.com/content/v257655u88w2/?sortorder=asc&p_o=20


See Also

summary.micro, plotMicro, valTable
**Examples**

```r
data(Tarragona)
m1 <- microaggregation(Tarragona, method="onedims", aggr=3)
## summary(m1)
data(testdata)
m2 <- microaggregation(testdata[1:100,c("expend","income","savings")],
  method="mdav", aggr=4)
summary(m2)
```

## for objects of class sdcMicro:

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

---

**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**

```r
microaggrGower(object, variables = NULL, aggr = 3, dist_var = NULL,
  by = NULL, mixed = NULL, mixed.constant = NULL, trace = FALSE,
  weights = NULL, numFun = mean, catFun = sampleCat, addrandom = FALSE)
```

**Arguments**

- **obj**: an object of class sdcMicroObj 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
TRUE/FALSE for some console output
numerical vector with length equal the number of variables for distance computation
function: to be used to aggregated numerical variables
function: to be used to aggregated categorical variables
TRUE/FALSE if a random value should be added for the distance computation.
a factor vector

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.

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

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

Alexander Kowarik

```r
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"))

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

sd <- microaggrGower(sdc)
```
Description

Small artificial toy data set.

Usage

data(microData)

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)
m1 <- microaggregation(microData, method="mdav")summary(m1)

plot.localSuppression

plot method for localSuppression objects

Description

Barplot for objects from class localSuppression.

Usage

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

Arguments

x object of class ‘localSuppression’

... Additional arguments passed through.

Details

Just look at the resulting plot.

Author(s)

Matthias Templ
See Also

localSuppression

Examples

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

plotMicro  

### Comparison plots

Description

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

Usage

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

Arguments

- **x**: object from class micro
- **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.

Author(s)

Matthias Templ

References


See Also

microaggregation
Examples

data(free1)
m1 <- microaggregation(free1[, 31:34], method="onedims", aggr=3)
m2 <- microaggregation(free1[, 31:34], method="pca", aggr=3)
plotMicro(m1, 0.1, which.plot=1)

---

Post Randomization

Description

To be used on categorical data. It randomly change the values of variables on selected records
(usually the risky ones) according to an invariant probability transition matrix.

Usage

pram(obj, variables=NULL, strata_variables=NULL, pd=0.8, alpha=0.5)
## S3 method for class 'pram'
print(x, ...)

Arguments

- obj: Input data. Allowed input data are objects of class 'matrix', 'data.frame', 'vector' or 'sdcMicroObj'.
- variables: Names of variables in 'obj' on which post-randomization should be applied. If
  obj is a vector, this argument is ignored.
- strata_variables: Names of variables for stratification (will be set automatically for an object of
  class 'sdcMicroObj'
- x: Output of pram()
- ...: further input, currently ignored.
- pd: minimum diagonal entries for the generated transition matrix P. Either a vector
  of length 1 or a vector of length ( number of categories ).
- alpha: amount of perturbation for the invariant Pram method

Value

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

Methods

signature(obj = "data.frame")
signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")
signature(obj = "ANY")
**Note**

The functionalities of pram_strata and pram are merged into pram, therefore pram_strata is deprecated.

**Author(s)**

Alexander Kowarik, Matthias Templ, Bernhard Meindl

**References**

http://www.gnu.org/software/glpk

**Examples**

```r
data(testdata)
res <- pram(testdata,
 variables=c("roof","sex"),
 strata_variables=c("urbrur","sex"))
print.pram(res)

res1 <- pram(testdata,variables=c("roof","walls","water"),strata_variables=c("urbrur","sex"))
print.pram(res1)
res2 <- pram(testdata,variables=c("roof","walls","water"),
 strata_variables=NULL)
print.pram(res2)

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

---

**print.freqCalc**

Print method for objects from class freqCalc

**Description**

Print method for objects from class freqCalc.

**Usage**

```r
## 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

dataHfrancdatI

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
print.localSuppression

See Also

indivrisk

Examples

```r
## 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:
indivrisk(f)
```

```
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

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

Print method for objects from class micro.

Usage

```r
## 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

```r
data(free1)
m1 <- microaggregation(free1[, 31:34], method="onedims", aggr=3)
m1
```

Description

Print method for objects from class suda2.

Usage

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

Arguments
x an object of class suda2

... additional arguments passed through.

Value
Table of dis suda scores.

Author(s)
Matthias Templ

See Also
suda2

Examples
## Not run:
data(testdata)
data_suda2 <- suda2(testdata,variables=c("urbrur","roof","walls","water","sex"))
data_suda2

## End(Not run)

---

**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=-1, R0=.95, P=0, missing=-999, seed=NULL)

Arguments
obj object of class sdcMicroObj or matrix or data frame
variables names or index of variables for that rank swapping is applied. For an object of class 'sdcMicroObj' 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 \(|X_1 - X_2| \leq \frac{2K₀X₁}{\sqrt{(N_S)}}\), 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 \(R₀ = \frac{R₁}{R₂}\) where \(R₁\) is the correlation coefficient of the two fields after swapping, and \(R₂\) 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  missig value code.

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. R₀ and K₀ are only used if positive. Only one of the two are used (R₀ is prefered if both are positive).

Value

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

Methods

signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")

Author(s)

Alexander Kowarik for the interface.

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

Examples

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

## 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 <- rankSwap(sdc)

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'
var  name of the variable(s) to be remove

Value
the modified “sdcMicroObj”

Methods
signature(obj = "sdcMicroObj")

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")
renameVars

Change the name of levels of a keyVariable in an object of class 'sdcMicroObj'

Description

Change the labels of levels.

Usage

renameVars(obj, var, before, after)

Arguments

obj object of class 'sdcMicroObj'
var name of the keyVariable to change
before vector of levels before
after vector of levels after

Value

the modified “sdcMicroObj”

Methods

signature(obj = "sdcMicroObj")

Examples

```r
## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2,
  numVars=c(‘expend’, ‘income’, ‘savings’), w=‘sampling_weight’)
sdc <- renameVars(sdc, var=“urbrur”, before=2, after=78)
```
Generate a HTML/LATEX output from an sdcMicroObj

Description

Summary statistics of the original and the perturbed data set

Usage

```r
report(obj,outdir=getwd(),filename="SDC-Report",format="HTML",title='SDC-Report',
internal=FALSE)
```

Arguments

- **obj**: an object of class 'sdcMicroObj' or 'reportObj'
- **outdir**: output folder
- **filename**: output filename
- **format**: HTML, TEXT or LATEX
- **title**: Title for the report
- **internal**: TRUE/FALSE, if TRUE a detailed internal report is produced, else a non-disclosive overview

Details

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

Author(s)

Matthias Templ, Bernhard Meindl

Examples

```r
## Not run:
data(testdata2)
sdc <- createSdcObj(testdata2,
   keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
   numVars=c('expend','income','savings'), w='sampling_weight')
report(sdc)
## End(Not run)
```
Description

These functions are provided for compatibility with older versions of the sdcMicro package only, and may be removed eventually. Commands that worked in versions of the sdcMicro package prior to version 3.1.1 will not necessarily work in version 3.1.2 and beyond, or may not work in the same manner.

Usage

```
localsupp2(x, keyVars, w, importance=rep(1, length(keyVars)),
method="minimizeSupp", k=1)
localsupp2Wrapper (x, keyVars, w, importance=rep(1, length(keyVars)),
method="minimizeSupp", kAnon=2)
pram_strata(obj, variables=NULL, strata_variables=NULL, pd=0.8, alpha=0.5)
```

Arguments

- `x` data frame or matrix
- `keyVars` column index of key variables
- `w` column index of sampling weights
- `importance` weights for each key variable
- `method` “minimizeSupp” (default), further methods will be included in future versions of the package
- `k` parameter for k-anonymity.
- `kAnon` parameter for k-anonymity.
- `obj` Input data. Allowed input data are objects of class 'matrix', 'data.frame', 'vector' or 'sdcMicroObj'.
- `variables` Names of variables in 'obj' on which post-randomization should be applied. If obj is a vector, this argument is ignored.
- `strata_variables` Names of variables for stratification (will be set automatically for an object of class 'sdcMicroObj'
- `pd` minimum diagonal entries for the generated transition matrix P. Either a vector of length 1 or a vector of length ( number of categories ).
- `alpha` amount of perturbation for the invariant Pram method

Details

localSupp2 is now a synonym for localSuppression. localSupp2Wrapper is now a synonym for localSuppression.
**Description**

Class to save all information about the SDC process

**Usage**

```r
createSdcObj(dat, keyVars, numVars = NULL, pramVars=NULL, weightVar = NULL, hhId = NULL, stratavar = NULL, sensibleVar=NULL, options = NULL)
undolast(obj)
nextSdcObj(obj)
show(object)
```

**Arguments**

- `dat` The microdata set. A numeric matrix or data frame containing the data.
- `obj` An object of class 'sdcMicroObj'
- `object` An object of class 'sdcMicroObj'
- `keyVars` Indices or names of categorical key variables. They must, of course, match with the columns of 'dat'.
- `pramVars` Indices or names of categorical variables considered to be pramed.
- `numVars` Index or names of continuous key 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)
- `options` additional options.

**Objects from the Class**

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

**Slots**

- `origData`: Object of class "dataframeOrNULL"~~
- `keyVars`: Object of class "numericOrNULL"~~
- `pramVars`: Object of class "numericOrNULL"~~
- `numVars`: Object of class "numericOrNULL"~~
- `weightVar`: Object of class "numericOrNULL"~~
- `hhId`: Object of class "numericOrNULL"~~
strataVar: Object of class "numericOrNULL" ~~
sensibleVar: Object of class "numericOrNULL" ~~
manipKeyVars: Object of class "dataframeOrNULL" ~~
manipPramVars: Object of class "dataframeOrNULL" ~~
manipNumVars: Object of class "dataframeOrNULL" ~~
manipStrataVar: Object of class "factorOrNULL" ~~
originalRisk: Object of class "listOrNULL" ~~
risk: Object of class "listOrNULL" ~~
utility: Object of class "listOrNULL" ~~
pram: Object of class "listOrNULL" ~~
localSuppression: Object of class "listOrNULL" ~~
options: Object of class "listOrNULL" ~~
additionalResults: Object of class "listOrNULL" ~~
set: Object of class "listOrNULL" ~~
prev: Object of class "sdcmicroOrNULL" ~~
deletedVars: Object of class "characterOrNULL" ~~

Methods

get.sdcMicroObj signature(object = "sdcMicroObj", type = "character"): ...
set.sdcMicroObj signature(object = "sdcMicroObj", type = "character", input = "listOrNULL"): ...
undo signature(object = "sdcMicroObj"): ...
nextSdcObj signature(object = "sdcMicroObj"): ...

Author(s)

Bernhard Meindl, Alexander Kowarik, Matthias Templ, Elias Rut

Examples

showClass("sdcMicroObj")
## Not run:
data(testdata)
sdc <- createSdcObj(testdata,
  keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
  numVars=c('expend','income','savings'), w='sampling_weight')
head(sdc@manipNumVars)
### Display Risks
sdc@risk$global
sdc <- dRisk(sdc)
sdc@risk$numeric
### use addNoize without Parameters
sdc <- addNoise(sdc,variables=c("expend","income"))
head(sdc@manipNumVars)
sdcMicroObj-class

sdc@risk$numeric
### undolast
sdc <- undolast(sdc)
head(sdc@manipNumVars)
sdc@risk$numeric
### redo addNoise with Parameter
sdc <- addNoise(sdc, noise=0.2)
head(sdc@manipNumVars)
sdc@risk$numeric
### dataGen
#sdc <- undolast(sdc)
#head(sdc@risk$individual)
#sdc@risk$global
#sdc <- dataGen(sdc)
#head(sdc@risk$individual)
#sdc@risk$global
### LocalSuppression
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
sdc <- localSuppression(sdc)
head(sdc@risk$individual)
sdc@risk$global
### microaggregation
sdc <- undolast(sdc)
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc <- microaggregation(sdc)
head(get.sdcMicroObj(sdc, type="manipNumVars"))
### pram
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
sdc <- pram(sdc,keyVar="water")
head(sdc@risk$individual)
sdc@risk$global
### pram_strata
sdc <- undolast(sdc)
head(pram_strata(sdc,variables=c("walls","water"))
head(sdc@risk$individual)
sdc@risk$global
### rankSwap
sdc <- undolast(sdc)
head(sdc@risk$individual)
sdc@risk$global
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc <- rankSwap(sdc)
head(get.sdcMicroObj(sdc, type="manipNumVars"))
head(sdc@risk$individual)
sdc@risk$global
\dontrun{
### suda2
sdc <- suda2(sdc)
sdc@risk$suda2
shuffle

### topBotCoding

```r
head(get.sdcMicroObj(sdc, type="manipNumVars"))
```

```r
sdc@risk$numeric
sdc <- topBotCoding(sdc, value=60000000, replacement=62000000, column="income")
head(get.sdcMicroObj(sdc, type="manipNumVars"))
```

```r
sdc@risk$numeric
### LocalRecProg
data(testdata2)
sdc <- createSdcObj(testdata2,
  keyVars=c("urbur", "roof", "walls", "water", "sex", "relat"))
```

```r
sdc@risk$global
sdc <- LocalRecProg(sdc)
sdc@risk$global
### llmodGlobalRisk
sdc <- undolast(sdc)
sdc <- llmodGlobalRisk(sdc, inclProb=0.001)
sdc@risk$model
```

```r
## End(Not run)
```

---

**shuffle**  

*Shuffling and EGADP*

---

**Description**

Data shuffling and General Additive Data Perturbation.

**Usage**

```r
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".
shuffle

```r
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 the perturbed values before shuffling is applied.

Value

If ‘obj’ is of class “sdcMicroObj” 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:

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

Methods

```r
signature(obj = "data.frame")
signature(obj = "matrix")
signature(obj = "sdcMicroObj")
```

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
References


See Also

rankSwap, lm

Examples

data(Prestige, package="car")
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)

suda2                Suda2: Detecting Special Uniques

Description

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

Usage

suda2(obj,...)#,variables=NULL,missing=-999,DisFraction=0.01)
Arguments

- **obj**: object of class “data.frame” or object of class “sdcMicroObj”
- **...**: 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.

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

Methods

- `signature(obj = "data.frame")`
- `signature(obj = "matrix")`
- `signature(obj = "sdcMicroObj")`

Author(s)

Alexander Kowarik 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


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


Examples

```r
## Not run:
data(testdata2)
data_suda2 <- suda(testdata2,variables=c("urbrur","roof","walls","water","sex"))
data_suda2
summary(data_suda2)

## 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 <- suda2(sdc)

## End(Not run)
```

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

Description

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

Usage

```r
## 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.
**summary.micro**

**Value**

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

**Author(s)**

Matthias Templ

**See Also**

freqCalc

**Examples**

```r
## 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$rk
## Local Suppression
localS <- localSupp(f, keyVar=2, indivRisk=indivf$rk, 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**

```r
## 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


See Also

microaggregation, valTable

Examples

data(Tarragona)
ml <- microaggregation(Tarragona, method="onedims", aggr=3)
## summary(ml)
**Summary method for objects from class pram**

### Description
Summary method for objects from class `pram` to provide information about transitions.

### Usage
```r
## 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

### References

### See Also
`pram`

### Examples
```r
data(free1)
x <- free1[, "MARSTAT"]
x2 <- pram(x)
x2
summary(x2)
```
swappNum

Rank Swapping

Description

Rank Swapping.

Usage

swappNum(x, w = 1:(dim(x)[2]), p)

Arguments

x matrix or data frame
w variables, on which rank swapping should be applied
p Percentage. Swapping range.

Details

The values of a variable are ranked, then each ranked value is swapped with another ranked value randomly chosen within a restricted range, i.e. the rank of two swapped values cannot differ by more than p percentile of the total number of records. The function apply the rank swapping on each variable independently.

Value

x original data
xm the rank swapped data
method info about the method name

Author(s)

Matthias Templ

References

Look, e.g. on http://www.niss.org/dgii/TR/dataswap-finalrevision.pdf

See Also

microaggregation

Examples

## Numerical Rank Swapping:
data(free1)
free1[, 31:34] <- rankSwap(free1[, 31:34], P=10)
Description

Rank Swapping.

Usage

swappNum(x, w = 1:(dim(x)[2]), p)

Arguments

x matrix or data frame
w variables, on which rank swapping should be applied
p Percentage. Swapping range.

Details

The values of a variable are ranked, then each ranked value is swapped with another ranked value randomly chosen within a restricted range, i.e. the rank of two swapped values cannot differ by more than p percentile of the total number of records. The function apply the rank swapping on each variable independently.

Value

x original data
xm the rank swapped data
method info about the method name

Author(s)

Matthias Templ

References

Look, e.g. on http://www.niss.org/dgii/TR/dataswap-finalrevision.pdf

See Also

microaggregation

Examples

## Numerical Rank Swapping:
data(free1)
free1[, 31:34] <- rankSwap(free1[, 31:34], P=10)
Tarragona data set

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

Usage
data(Tarragona)

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

Examples
data(Tarragona)
head(Tarragona)
dim(Tarragona)
A real-world data set on household income and expenditures

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

Usage
data(testdata)
data(testdata2)

Format
A data frame with 4580 observations on the following 14 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

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

```r  
data(testdata)  
## maybe str(testdata) ; plot(testdata) ...
```

topBotCoding  

**Top and Bottom Coding**

Description

Function for Top and Bottom Coding.

Usage

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

Arguments

- **obj**: vector or one-dimensional matrix or data.frame or object of class “sdcMicroObj”  
- **value**: limit, from where it should be top- or bottom-coded  
- **replacement**: replacement value.  
- **kind**: top or bottom  
- **column**: xxx

Details

Extreme values are replaced by one value to reduce the disclosure risk.
valTable

Value

Top or bottom coded data or modified “sdcMicroObj”.

Methods

signature(obj = "ANY")
signature(obj = "sdcMicroObj")

Author(s)

Matthias Templ

See Also

indivRisk

Examples

data(free1)
topBotCoding(free1[,"DEBTS"], value=9000, replacement=9100, kind="top")

## 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=50000, replacement=1000, column="income")
testdataout <- extractManipData(sdc)

valTable

Comparison of different microaggregation methods

Description

A Function for the comparison of different perturbation methods.

Usage

call(valTable(x, 
    method=c("simple","onedims","clustppca","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: data frame or matrix
method: microaggregation methods or adding noise methods or rank swapping.
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

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

Value

Measures of information loss split by the comparison of different methods.

Methods for adding noise should be named via “addNoise: method”, e.g. “addNoise: correlated”, i.e. the term ‘at first’ then followed by a ‘:’ and a blank and then followed by the name of the method as described in function ‘addNoise’.

Author(s)

Matthias Templ

References


See Also

microaggregation, summary.micro

Examples

data(Tarragona)
## Not run:
valTable(Tarragona[100:200,],
method=c("simple","onedims","pca","addNoise: additive"))
valTable(Tarragona,
method=c("simple","onedims","pca","clustpcppca",

valTable
"mdav", "addNoise: additive", "swappNum")
## clustpppca in combination with Mclust outperforms
## the other algorithms for this data set...

## End(Not run)

---

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

**Description**

Change the scale of a variable

**Usage**

```r
varToNumeric(obj, var)
varToFactor(obj, var)
```

**Arguments**

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

**Value**

the modified "sdcMicroObj"

**Methods**

- `signature(obj = "sdcMicroObj")`

**Examples**

```r
## 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")
```
Index

*Topic **aplot**
  plot.localSuppression, 50
  plotMicro, 51

*Topic **classes**
  freq-methods, 24
  sdcMicroObj-class, 63

*Topic **datasets**
  casc1, 11
  CASCrefermicrodata, 12
  EIA, 19
  francdat, 23
  free1, 24
  microData, 50
  Tarragona, 76
  testData, 77

*Topic **manip**
  addNoise, 8
  dataGen, 13
  dRisk, 14
  dRiskRMD, 15
  dUtility, 17
  freqCalc, 25
  globalRecode, 28
  indivRisk, 30
  LLmodGlobalRisk, 31
  LocalRecProg, 33
  localSupp, 34
  localSupp2, 36
  localSupp2Wrapper, 38
  localSuppression, 39
  mafast, 40
  measure_risk, 42
  microaggregation, 45
  pram, 52
  renameVars, 60
  shuffle, 66
  suda2, 68
  swappNum, 74
  swappNum-deprecated, 75
  topBotCoding, 78

*Topic **methods**
  calcRisks, 10
  extractManipData, 22
  groupVars, 29
  removeDirectID, 59
  report, 61
  varToFactor, 81

*Topic **package**
  sdcMicro-package, 3

*Topic **print**
  print.freqCalc, 53
  print.indivRisk, 54
  print.localSuppression, 55
  print.micro, 56
  print.suda2, 56
  summary.freqCalc, 70
  summary.micro, 71
  summary.pram, 73
  valTable, 79

  addNoise, 8
  addNoise, data.frame-method (addNoise), 8
  addNoise, matrix-method (addNoise), 8
  addNoise, sdcMicroObj-method (addNoise), 8
  addNoise-methods (addNoise), 8

  calcRisks, 10
  calcRisks, sdcMicroObj-method
    (calcRisks), 10
  calcRisks-methods (calcRisks), 10
  casc1, 11
  CASCrefermicrodata, 12
  createSdcObj (sdcMicroObj-class), 63
  cut, 29

  dataGen, 13
  dataGen, data.frame-method (dataGen), 13
  dataGen, matrix-method (dataGen), 13
INDEX

dataGen, sdcMicroObj-method (dataGen), 13
dataGen-methods (dataGen), 13
dRisk, 14, 17, 18
dRisk, data.frame-method (dRisk), 14
dRisk, matrix-method (dRisk), 14
dRisk, sdcMicroObj-method (dRisk), 14
dRisk-methods (dRisk), 14
dRiskRMD, 15, 18
dRiskRMD, data.frame-method (dRiskRMD), 15
dRiskRMD, matrix-method (dRiskRMD), 15
dRiskRMD, sdcMicroObj-method (dRiskRMD), 15
dRiskRMD-methods (dRiskRMD), 15
dUtility, 15, 17
dUtility, data.frame-method (dUtility), 17
dUtility, matrix-method (dUtility), 17
dUtility, sdcMicroObj-method (dUtility), 17
dUtility-methods (dUtility), 17
EIA, 19
extractManipData, 22
extractManipData, sdcMicroObj-method
(extractManipData), 22
extractManipData-methods
(extractManipData), 22
francdat, 23
free1, 24
dfreq (freq-methods), 24
dfreq, sdcMicroObj-method (freq-methods), 24
dfreq-methods, 24
dfreqCalc, 25, 31, 35, 37, 39, 44, 54, 71
generateStrata, 27
get.sdcMicroObj (sdcMicroObj-class), 63
get.sdcMicroObj, sdcMicroObj, character-method
(sdcMicroObj-class), 63
globalRecode, 28
globalRecode, ANY-method (globalRecode), 28
globalRecode, sdcMicroObj-method
(globalRecode), 28
globalRecode-methods (globalRecode), 28
groupVars, 29
groupVars, sdcMicroObj-method
(groupVars), 29
groupVars-methods (groupVars), 29
indivRisk, 27, 30, 35, 44, 55, 79
ldiversity (measure_risk), 42
ldiversity, data.frame-method
(measure_risk), 42
ldiversity, matrix-method
(measure_risk), 42
ldiversity, sdcMicroObj-method
(measure_risk), 42
ldiversity-methods (measure_risk), 42
LLmodGlobalRisk, 31
LLmodGlobalRisk, ANY-method
(LLmodGlobalRisk), 31
LLmodGlobalRisk, data.frame-method
(LLmodGlobalRisk), 31
LLmodGlobalRisk, matrix-method
(LLmodGlobalRisk), 31
LLmodGlobalRisk, sdcMicroObj-method
(LLmodGlobalRisk), 31
LLmodGlobalRisk-methods
(LLmodGlobalRisk), 31
lm, 68
LocalRecProg, 33
LocalRecProg, data.frame-method
(LocalRecProg), 33
LocalRecProg, matrix-method
(LocalRecProg), 33
LocalRecProg, sdcMicroObj-method
(LocalRecProg), 33
LocalRecProg-methods (LocalRecProg), 33
localSupp, 34
localSupp, ANY-method (localSupp), 34
localSupp, sdcMicroObj-method
(localSupp), 34
localSupp-methods (localSupp), 34
localSupp2, 36
localSupp2Wrapper, 38
localSuppression, 37, 39, 39, 51, 55, 62
localSuppression, data.frame-method
(localSuppression), 39
localSuppression, matrix-method
(localSuppression), 39
localSuppression, sdcMicroObj-method
(localSuppression), 39
localsuppression-methods
(localSuppression), 39
loglm, 32
mafast, 40
mafast, ANY-method (mafast), 40
mafast, data.frame-method (mafast), 40
mafast, matrix-method (mafast), 40
mafast, sdcMicroObj-method (mafast), 40
mafast-methods (mafast), 40
maxCat (microaggrGower), 48
measure_risk, 27, 31, 32, 42
measure_risk, data.frame-method
(measure_risk), 42
measure_risk, matrix-method
(measure_risk), 42
measure_risk, sdcMicroObj-method
(measure_risk), 42
measure_risk-methods (measure_risk), 42
microaggregation, 41, 45, 51, 56, 72, 74, 75, 80
microaggregation, ANY-method
(microaggregation), 45
microaggregation, data.frame-method
(microaggregation), 45
microaggregation, matrix-method
(microaggregation), 45
microaggregation, sdcMicroObj-method
(microaggregation), 45
microaggregation-methods
(microaggregation), 45
microaggrGower, 48
microaggrGower, data.frame-method
(microaggrGower), 48
microaggrGower, sdcMicroObj-method
(microaggrGower), 48
microaggrGower-methods
(microaggrGower), 48
microData, 50
nextSdcObj (sdcMicroObj-class), 63
nextSdcObj, sdcMicroObj-method
(sdcMicroObj-class), 63
plot.localSuppression, 50
plotMicro, 47, 51
pram, 52, 73
pram, data.frame-method (pram), 52
pram, matrix-method (pram), 52
pram, sdcMicroObj-method (pram), 52
pram, vector-method (pram), 52
pram-methods (pram), 52
pram_strata (sdcMicro-deprecated), 62
pram_strata, ANY-method
(sdcMicro-deprecated), 62
print (freq-methods), 24
print, sdcMicroObj-method
(freq-methods), 24
print-methods (freq-methods), 24
print.freqCalc, 53
print.indivRisk, 54
print.ldrdiversity (measure_risk), 42
print.localSuppression, 55
print.measure_risk (measure_risk), 42
print.micro, 56
print.pram (pram), 52
print.sudaR, 56
rankSwap, 57, 68
rankSwap, data.frame-method (rankSwap), 57
rankSwap, matrix-method (rankSwap), 57
rankSwap, sdcMicroObj-method (rankSwap), 57
rankSwap-methods (rankSwap), 57
removeDirectID, 59
removeDirectID, sdcMicroObj-method
(removeDirectID), 59
removeDirectID-methods
(removeDirectID), 59
renameVars, 60
renameVars, sdcMicroObj-method
(renameVars), 60
renameVars-methods (renameVars), 60
report, 61
report, sdcMicroObj-method (report), 61
report-methods (report), 61
sampleCat (microaggrGower), 48
sdcMicro (sdcMicro-package), 3
sdcMicro-deprecated, 62
sdcMicro-package, 3
sdcMicroObj-class, 63
set.sdcMicroObj (sdcMicroObj-class), 63
set.sdcMicroObj, sdcMicroObj, character, listOrNULL-method
(sdcMicroObj-class), 63
show (sdcMicroObj-class), 63
show, sdcMicroObj-method
  (sdcMicroObj-class), 63
shuffle, 14, 66
shuffle, data.frame-method (shuffle), 66
shuffle, matrix-method (shuffle), 66
shuffle, sdcMicroObj-method (shuffle), 66
shuffle-methods (shuffle), 66
suda2. 57, 68
suda2, data.frame-method (suda2), 68
suda2, matrix-method (suda2), 68
suda2, sdcMicroObj-method (suda2), 68
suda2-methods (suda2), 68
summary.freqCalc, 70
summary.micro, 10, 47, 71, 80
summary.pram, 73
swappNum, 74
swappNum-deprecated, 75

Tarragona, 76
testdata, 77
testdata2 (testdata), 77
topBotCoding, 78
topBotCoding, ANY-method (topBotCoding), 78
topBotCoding, sdcMicroObj-method
  (topBotCoding), 78
topBotCoding-methods (topBotCoding), 78

undolast (sdcMicroObj-class), 63
undolast, sdcMicroObj-method
  (sdcMicroObj-class), 63

valTable, 47, 72, 79
varToFactor, 81
varToFactor, sdcMicroObj-method
  (varToFactor), 81
varToFactor-methods (varToFactor), 81
varToNumeric (varToFactor), 81
varToNumeric, sdcMicroObj-method
  (varToFactor), 81
varToNumeric-methods (varToFactor), 81