Package ‘sdcMicro’

Type Package

Title Statistical Disclosure Control methods for the generation of public- and scientific-use files.

Version 4.0.0

Date 2013-03-30

Author Matthias Templ, Alexander Kowarik, Bernhard Meindl

Maintainer Matthias Templ <matthias.templ@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 generation of public- and scientific-use files. The package sdcMicroGUI includes a graphical user interface for the methods in this package.

Depends R (>= 2.10), R2HTML

Imports car, robustbase, cluster, MASS, e1071, R2HTML, Rcpp, Rglpk, methods

License GPL-2

Collate '0classes.r' 'addNoise.r' 'aux_functions.r' 'dataGen.r' 'dRisk.R' 'dRiskRMD.R' 'dUtility.R' 'freqCalc.r' 'globalRecode.R' ...

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-03-22 10:44:39

R topics documented:

  sdcMicro-package .......................................................... 3
  addNoise ................................................................. 7
  calcRisks .................................................................. 9
  casc1 ................................................................. 10
  CASCrefmicrodata ........................................................ 11
R topics documented:

- dataGen .................................................. 12
- dRisk ....................................................... 13
- dRiskRMD ................................................... 14
- dUtility ....................................................... 16
- EIA .......................................................... 18
- francdat .................................................... 20
- free1 ......................................................... 21
- freqCalc ..................................................... 22
- globalRecode .............................................. 24
- groupVars .................................................. 25
- indivRisk ................................................... 26
- LLmodGlobalRisk ......................................... 27
- LocalRecProg ............................................. 28
- localSupp .................................................. 30
- localSupp2 ............................................... 31
- localSupp2Wrapper ...................................... 33
- localSuppression ........................................ 34
- measure_risk ............................................. 36
- microaggregation ....................................... 38
- microData .................................................. 41
- plot.localSuppression ................................... 41
- plotMicro .................................................. 42
- pram ......................................................... 43
- pram_strata ............................................... 45
- print.freqCalc .......................................... 46
- print.indivRisk .......................................... 47
- print.localSuppression ................................ 48
- print.micro ............................................... 49
- print.pram ................................................ 49
- print.suda2 .............................................. 50
- rankSwap .................................................. 51
- renameVars ................................................ 52
- report ....................................................... 53
- sdcMicro-deprecated ..................................... 54
- sdcMicroObj-class ....................................... 55
- shuffle ...................................................... 58
- suda2 ....................................................... 60
- summary.freqCalc ....................................... 62
- summary.micro .......................................... 63
- summary.pram .......................................... 64
- swappNum ................................................ 65
- swappNum-deprecated ................................... 66
- Tarragona .................................................. 67
- testdata .................................................... 68
- topBotCoding ............................................. 70
- valTable .................................................... 71
- varToFactor .............................................. 72

Index ......................................................... 74
**sdcMicro-package**

**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 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.

Please note, that methods “shuffling”, “robShuffle” (robust shuffling), “gadp” and “robgadp” are not included in the package because method “shuffling” is under a US-patent by other authors, even shuffling consists only of 8 lines of code . . .

**Details**

<table>
<thead>
<tr>
<th>Package:</th>
<th>sdcMicro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>2.5.9</td>
</tr>
<tr>
<td>Date:</td>
<td>2009-07-22</td>
</tr>
<tr>
<td>License:</td>
<td>GPL 2.0</td>
</tr>
</tbody>
</table>

**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$fk f$Fk
## with missings:
x <- francdat
x[3,5] <- NA
x[4,2] <- x[4,4] <- NA
x[5,6] <- NA
x[6,2] <- NA
f2 <- freqCalc(x, keyVars=c(2,4,5,6),w=8)
f2$Fk
## individual risk calculation:
indivf <- indivRisk(f)
indivf$rk
## Local Suppression
localS <- localSupp(f, keyVar=2, indivRisk=indivf$rk, threshold=0.25)
f2 <- freqCalc(localS$freqCalc, keyVars=c(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=3)
ind <- indivRisk(f)
## and now you can use the interactive plot for individual risk objects:
## plot(ind)

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

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

## 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('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 addNoise without Parameters
sdc <- addNoise(sdc,variables=c("expend","income"))
head(sdc@manipNumVars)
### 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"))
addNoise

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)
head(sdc@risk$individual)
sdc@risk$global
### suda2
sdc <- suda2(sdc)
head(sdc@risk$suda2)
### topBotCoding
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc <- topBotCoding(sdc, value=6/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash, replacement=62/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash, column="income")
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc@risk$numeric
### LocalRecProg
data(testdata2)
sdc <- createSdcObj(testdata2, keyVars=c("urbrur", "roof", "walls", "water", "sex", "relat"))
sdc@risk$global
sdc <- LocalRecProg(sdc)
sdc@risk$global
### LLmodGlobalRisk
sdc <- undolast(sdc)
sdc <- LLmodGlobalRisk(sdc, inclProb=1)
sdc@risk$model

## End(Not run)

---

**Description**

Various methods for adding noise to perturb continuous scaled variables.
addNoise

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

Arguments

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

Details
Method ‘additive’ adds noise completely at random to each variable depending on the 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
An object of class “micro” with following entities:
x the original data
xm the modified (perturbed) data
method method used for perturbation
noise amount of noise

Methods
signature(obj = “data.frame”)signature(obj = “matrix”)signature(obj = “sdcMicroObj”)
References


See Also

summary.micro

Examples

data(Tarragona)
a1 <- addNoise(Tarragona)
a1
data(testdata)
testdata[, c('expend','income','savings')] <- addNoise(testdata[,c('expend','income','savings')])$xm

# 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 <- addNoise(sdc)

calcRisks

Recalculate 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")

Examples

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

casc1

Small Artificial Data set

Description

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

Usage

data(casc1)

Format

The format is: int [1:13, 1:7] 10 12 17 21 9 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.

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

Source

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

References


Examples

data(CASCrefmicrodata)
str(CASCrefmicrodata)
dataGen

Fast generation of synthetic data

Description

Fast generation of (primitive) synthetic multivariate normal data.

Usage

```r
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

Examples

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

```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 <- dataGen(sdc)
```

dRisk  

**overall disclosure risk**

Description

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

Usage

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

Arguments

- `obj` original data
- `...` 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.

Methods

- `signature(obj = "data.frame")`
- `signature(obj = "matrix")`
- `signature(obj = "sdcMicroObj")`
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
... see possible arguments below
xm masked data
k weight for adjusting the influence of the robust Mahalanobis distances, i.e. to increase or decrease each of the disclosure risk intervals.
k2 parameter for method RMDID2 to choose a small interval around each masked observation.
Details

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

The algorithm works as follows:
1. Robust Mahalanobis distances are estimated in order to get a robust multivariate distance for each observation.
2. Intervals are estimated for each observation around every data point of the original data points where the length of the interval is defined/weighted by the squared robust Mahalanobis distance and the parameter $k$. The higher the RMD of an observation the larger the interval.
3. Check if the corresponding masked values fall into the intervals around the original values or not. If the value of the corresponding observation is within such an interval the whole observation is considered unsafe. So, we get a whole vector indicating which observation is save or not, and we are finished already when using method RMDID1).
4. For method RMDID1w: we return the weighted (via RMD) vector of disclosure risk.
5. For method RMDID2: whenever an observation is considered unsafe it is checked if $m$ other observations from the masked data are very close (defined by a parameter $k2$ for the length of the intervals as for SDID or RSDID) to such an unsafe observation from the masked data, using Euclidean distances. If more than $m$ points are in such a small interval, we conclude that this observation is “save”.

Value

The disclosure risk.

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

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

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

---

**dUtility**

**Data utility**

**Description**

IL1s data utility.

**Usage**

```r
dUtility(obj,...)#, xm, method="IL1")
```

**Arguments**

- **obj**: original data
- **...**: 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

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/lncs30500utlier.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, xm=y)
  dRisk(x, xm=y)
  dUtility(x, xm=y)
  dUtility(x, xm=y, method="eigen")
  dUtility(x, xm=y, method="robeigen")

## for objects of class sdcMicro:
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.

<table>
<thead>
<tr>
<th>UTILITYID</th>
<th>UNIQUE UTILITY IDENTIFICATION NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTILNAME</td>
<td>UTILITY NAME. A factor with levels 4-County Electric Power Assn Alabama Power Co</td>
</tr>
<tr>
<td></td>
<td>Alaska Electric Light &amp; Power Co Anchoraghe City of Anoka Electric Coop Appalachian Electric Coop</td>
</tr>
<tr>
<td></td>
<td>Appalachian Power Co Arizona Public Service Co Arkansas Power &amp; Light Co Arkansas Valley Electric Co</td>
</tr>
<tr>
<td></td>
<td>Atlantic City Electric Company Baker Electric Coop Inc Baltimore Gas &amp; Electric Co</td>
</tr>
<tr>
<td></td>
<td>Bangor Hydro-Electric Co Berkeley Electric Coop Inc Black Hills Corp Blackstone Valley Electric Co</td>
</tr>
<tr>
<td></td>
<td>Bonneville Power Admin Boston Edison Co Bountiful City Light &amp; Power Bristol City of</td>
</tr>
<tr>
<td></td>
<td>Brookings City of Brunswick Electric Member Corp Burlington City of Carolina Power &amp; Light Co</td>
</tr>
<tr>
<td></td>
<td>Carroll Electric Coop Corp Cass County Electric Coop Inc Central Illinois Light Company</td>
</tr>
<tr>
<td></td>
<td>Central Illinois Pub Serv Co Central Louisiana Elec Co Inc Central Maine Power Co</td>
</tr>
<tr>
<td></td>
<td>Central Power &amp; Light Co Central Vermont Pub Serv Corp Chattanooga City of Cheyenne Light Fuel &amp;</td>
</tr>
<tr>
<td></td>
<td>Chugach Electric Assn Inc Cincinnati Gas &amp; Electric Co Citizens Utilities Company</td>
</tr>
<tr>
<td></td>
<td>City of Boulder City of Clinton City of Dover City of Eugene City of Gillette</td>
</tr>
<tr>
<td></td>
<td>City of Groton Dept of Utilities City of Idaho Falls City of Independence City of Newark</td>
</tr>
<tr>
<td></td>
<td>City of Reading City of Tupelo Water &amp; Light D Clarksville City of Cleveland City of</td>
</tr>
<tr>
<td></td>
<td>Cleveland Electric Illum Co Coast Electric Power Assn Cobb Electric Membership Corp</td>
</tr>
<tr>
<td></td>
<td>Colorado River Commission Colorado Springs City of Columbus Southern Power Co</td>
</tr>
<tr>
<td></td>
<td>Commonwealth Edison Co Connecticut Electric Co Connecticut Light &amp; Power Co</td>
</tr>
<tr>
<td></td>
<td>Consolidated Edison Co NY Inc Consumers Power Co Cornhusker Public Power Dist</td>
</tr>
<tr>
<td></td>
<td>Cuivre River Co-Op Cumberland Elec Member Corp Dakota Electric Assn</td>
</tr>
<tr>
<td></td>
<td>Dawson County Public Pwr Dist Dayton Power &amp; Light Company Decatur City of Delaware Electric Coop</td>
</tr>
<tr>
<td></td>
<td>Delmarva Power &amp; Light Co Detroit Edison Co Duck River Elec Member Corp Duke Power Co</td>
</tr>
<tr>
<td></td>
<td>Duquesne Light Company East Central Electric Assn Eastern Maine Electric Coop</td>
</tr>
<tr>
<td></td>
<td>Fairbanks City of Fayetteville Public Works Comm First Electric Coop Corp Florence City of</td>
</tr>
<tr>
<td></td>
<td>Florida Power &amp; Light Co Florida Power Corp Fort Collins Lgt &amp; Pwr Utility Fremont City of</td>
</tr>
</tbody>
</table>
Georgia Power Co Gibson County Elec Member Corp Golden Valley Elec Assn Inc
Grand Island City of Granite State Electric Co Green Mountain Power Corp Green River Electric Corp
Greeneville City of Gulf Power Company Gulf States Utilities Co Hasting Utilities
Hawaii Electric Light Co Inc Hawaiian Electric Co Inc Henderson-Union Rural E C C
Homer Electric Assn Inc Hot Springs Rural El Assn Inc Houston Lighting & Power Co
Huntsville City of Idaho Power Co IES Utilities Inc Illinois Power Co Indiana Michigan Power Co
Indianapolis Power & Light Co Intermountain Rural Elec Assn Interstate Power Co
Jackson Electric Member Corp Jersey Central Power & Light Co Joe Wheeler Elec Member Corp
Johnson City City of Jones-Onslow Elec Member Corp Kansas City City of Kansas City Power & Light Co
Kentucky Power Co Kentucky Utilities Co Ketchikan Public Utilities Kingsport Power Co
Knoxville City of Kodiak Electric Assn Inc Kootenai Electric Coop, Inc Incl Lansing Board of Water & Light
Lenoir City City of Lincoln City of Long Island Lighting Co Los Angeles City of
Louisiana Power & Light Co Louisville Gas & Electric Co Loup River Public Power Dist
Lower Valley Power & Light Inc Maine Public Service Company Massachusetts Electric Co
Matanuska Electric Assn Inc Maui Electric Co Ltd McKenzie Electric Coop Inc
Memphis City of MidAmerican Energy Company Middle Tennessee E M Co Midwest Energy, Inc
Minnesota Power & Light Co Mississippi Power & Light Co Minnesota Power Co
Monongahela Power Co Montana-Dakota Utilities Co Montana Power Co Moon Lake Electric Assn Inc
Narragansett Electric Co Inc 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 Podak 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 Co Oklahoma Electric Co Oper Inc Oklahoma Gas & Electric Co
Ohio-Mercer Elec Coop Inc Omaha Public Power District Otter Tail Power Co
Pacific Gas & Electric Co Pacificorp dba Pacific Pwr & L Palmetto Electric Coop, Inc
Pennsylvania Power & Light Co Pennyrile Rural Electric Coop Philadelphia Electric Co
Poudre Valley REA Inc Power Authority of State of NY Provo City Corporation
Public Service Co of Colorado Public Service Co of IN Inc Public Service Co of NH
Public Service Co of NM Public Service Co of Oklahoma Public Service Electric & Gas Co
PUD No 1 of Clark County PUD No 1 of Snohomish County Puget Sound Power & Light Co
Rappahannock Electric Coop Rochester Public Utilities Rockland Electric Company
Rosebud Electric Coop Inc Rutherford Elec Member Corp Sacramento Municipal Util Dist
Salmon River Electric Coop Inc Salt River Proj Ag I & P Dist San Antonio City of
Sioux Valley Empire E A Inc South Carolina Electric & Gas Co South Carolina Pub Serv Auth
South Kentucky Rural E C Co Southern California Edison Co Southern Nebraska Rural P P D
Southern Pine Electric Power Assn Southwest Tennessee E M Co Southern Western Electric Power Co
Southwestern Public Service Co Springfield City of St Joseph Light & Power Co
State Level Adjustment Tacoma City of Tampa Electric Co Texas-New Mexico Power Co
Texas Utilities Electric Co Tri-County Electric Assn Inc Tucson Electric Power Co
Turner-Hutchinson EL Coop, Inc TVA U S Bureau of Indian Affairs Union Electric Co
Union Light & Power Co United Illuminating Co Upper Cumberland E M Co UtiliCorp United Inc
Verdigris Valley Electric Coop Verendrye Electric Coop Inc Virginia Electric & Power Co
Volunteer Electric Coop Wallingford Town of Warren Rural Elec Coop Corp Washington Water Power Co
Watertown Municipal Util Dept Wells Rural Electric Co West Penn Power Co West Plains Electric Co
West River Electric Assn Inc Western Massachusetts Elec Co Western Resources Inc
Wheeling Power Company
Wisconsin Electric Power Co
Wisconsin Power & Light Co
Wisconsin Public Service Corp
Wright-Hennepin Coop Elec Assn
Yellowstone Vlly Elec Coop Inc

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

YEAR  REPORTING YEAR FOR THE DATA

MONTH  REPORTING MONTH FOR THE DATA

RESREVENUE  REVENUE FROM SALES TO RESIDENTIAL CONSUMERS
RESSALES  SALES TO RESIDENTIAL CONSUMERS

COMREVENUE  REVENUE FROM SALES TO COMMERCIAL CONSUMERS
COMSALES  SALES TO COMMERCIAL CONSUMERS

INDREVENUE  REVENUE FROM SALES TO INDUSTRIAL CONSUMERS
INDSALES  SALES TO INDUSTRIAL CONSUMERS

OTHREVENUE  REVENUE FROM SALES TO OTHER CONSUMERS
OTHRSALES  SALES TO OTHER CONSUMERS

TOTREVENUE  REVENUE FROM SALES TO ALL CONSUMERS
TOTSFALES  SALES TO ALL CONSUMERS

Source

Public use file from the CASC project.

References


Examples

data(francdat)

Description

Small synthetic data from Capobianchi, Polettini, Lucarelli

Usage

data(francdat)
**free1**

**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**

```r
data(francdat)
francdat
```

---

**Description**

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

**Usage**

```r
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)

freqCalc

Frequencies calculation for risk estimation

Description

Fast computation and estimation of the sample and population frequency counts which is also needed for risk estimation.

Usage

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.

Note that function measure_risk() also estimates the frequencies, and this function using an faster C++ code. freqCalc() does not support sdcMicro S4 class objects.
freqCalc

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

look e.g. in http://neon.vb.cbs.nl/casc/Deliv/12d1.pdf Templ, M. Statistical Disclosure
Control for Microdata Using the R Package sdcMicro, Transactions on Data Privacy, vol. 1, number

Templ, M. New Developments in Statistical Disclosure Control and Imputation: Robust Statis-
3838108280, 264 pages.

and Anonymity in Information Management Systems New Techniques for New Practical Problems,

See Also

indivRisk, measure_risk

Examples

data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$freqCalc
f$fk
f$Fk
## with missings:
x <- francdat
x[3,5] <- NA
x[4,2] <- x[4,4] <- NA
x[5,6] <- NA
x[6,2] <- NA
f2 <- freqCalc(x, keyVars=c(2,4,5,6),w=8)
f2$Fk

# 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) )

globalRecode

Global Reoding

Description
Global recoding

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

Arguments

obj vector of class numeric or of class factor with integer labels for recoding
column which keyVar should be changed
... see possible arguments below
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 approximately 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
A factor is returned, unless labels = FALSE which results in the mere integer level codes.

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

```r
groupVars(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

### Methods

```r
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)
```

<table>
<thead>
<tr>
<th>indivRisk</th>
<th>Individual Risk computation</th>
</tr>
</thead>
</table>

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**


**See Also**

delay, 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
```

---

**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**

```r
LLmodGlobalRisk(obj,...)
```

# LLmodGlobalRisk(x, method = "IPF", inclProb = NULL, form = as.formula(paste(" ~ ", paste(colnames(x), collapse = "+"), sep = " ")))

**Arguments**

- **obj**: A numeric matrix or data frame containing the categorical key variables.
- **...**: see possible arguments below
- **method**: At this time, only iterative proportional fitting (“IPF”) can be used.
- **x**: data frame or matrix
- **inclProb**: Inclusion probabilities (experimental)
- **form**: A formula specifying the model.
- **modOutput**: if TRUE, additional output is given.
Details

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

Value

Two global risk measures.

Author(s)

Matthias Templ

References


See Also

loglm, measure_risk

Examples

data(free1)
x <- data.frame(free1[,c(2,4:5)])
x["SEX"] <- as.factor(x[,"SEX"])
x["MARSTAT"] <- as.factor(x[,"MARSTAT"])
x["KINDPERS"] <- as.factor(x[,"KINDPERS"])
LLmodGlobalRisk(x, inclProb=1/mean(free1[,"WEIGHT"]))

## 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 <- LLmodGlobalRisk(sdc)

LocalRecProg

Local recoding via Edmond’s maximum weighted matching algorithm

Description

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

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

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

Arguments

obj             Input data
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 suppressed variables (suffix _lr).

Methods

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

Author(s)

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

References

localSupp

Local Suppression

Description
A simple method to perform local suppression.

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

Arguments
obj          object from class freqCalc
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

Methods
signature(obj = "ANY")
signature(obj = "sdcMicroObj")
localSupp2

Author(s)
Matthias Templ

References

See Also
dfreqCalc, indivRisk

Examples

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

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

## Local Suppression
localS <- localSupp(f, keyVar=2, indivRisk=indivf$rk, threshold=0.25)
f2 <- freqCalc(localS$freqCalc, keyVars=c(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

## 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 <- localSupp(sdc, keyVar='urbrur')

localSupp2 Local Suppression 2

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

Usage

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 preferred 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 alorithm 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-anonymizated 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 (sligthly).

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
localSupp2Wrapper

References


See Also
deci, localSuppression

Examples

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

localSupp2Wrapper Local Suppression 2

Description

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

Usage

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.
localSuppression

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,...)#, keyVars, k=2, importance=NULL)
**localSuppression**

### Arguments

- **obj**  
  data frame or matrix
- **...**  
  see arguments below
- **keyVars**  
  numeric vector specifying indices of (categorical) key-variables
- **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.

### 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

### Methods

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

### Author(s)

Bernhard Meindl, Matthias Templ

### Examples

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

## 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 <- localSuppression(sdc)
```
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.

```r
measure_risk(obj,...)
#measure_risk(data,keyVars,w=NULL,missing=-999,hid=NULL,max_global_risk=0.01,fast_hier=TRUE)
ldiversity(data,keyVars,missing=-999,l_recurs_c=2,ldiv_index=NULL)
## 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 Household ID
- **missing**: A integer value to be used as missing value in the C++ routine
- **l_recurs_c**: L-Diversity Constant
- **ldiv_index**: indices (or names) of the variables used for l-diversity
- **x**: Output of `measure_risk`, `measure_hier` or `measure_thres`
- **max_global_risk**: Maximal global risk for threshold computation
- **fast_hier**: If TRUE a faster approximation is computed if household data are provided.
Value

- **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


See Also

- `freqCalc`, `indivRisk`

Examples

```r
## 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...
slotNames(sdc@risk)
```

```r
## measure risk on data frames or matrices:
```
res <- measure_risk(testdata,c("urbrur", "roof", "walls", "water", "sex"))
print(res)
head(res$Res)
resw <- measure_risk(testdata,c("urbrur", "roof", "walls", "water", "sex"), w="sampling_weight")
print(resw)
head(resw$Res)
res1 <- ldiversity(testdata,c("urbrur", "roof", "walls", "water", "sex"), ldiv_index="electcon")
print(res1)
head(res1)
res2 <- ldiversity(testdata,c("urbrur", "roof", "walls", "water", "sex"), ldiv_index=c("electcon", "relat"))
print(res2)
head(res2)
# measure risk with household risk
resh <- measure_risk(testdata,c("urbrur", "roof", "walls", "water", "sex"), w="sampling_weight", hid="ori_hid")
print(resh)
# change max_global_risk
rest <- measure_risk(testdata,c("urbrur", "roof", "walls", "water", "sex"), w="sampling_weight", max_global_risk=0)
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 availabe in object sdc:
sdc <- measure_risk(sdc)

microaggregation

**Microaggregation**

**Description**

Function to perform various methods of microaggregation.

**Usage**

microaggregation(obj, variables, aggr=3, strata_variables=NULL, ...)#, method = "pca", weights=NULL, nc =

**Arguments**

- **obj**: data frame or matrix
- **variables**: variables to microaggregate
- **aggr**: aggregation level (default=3)
- **strata_variables**: by-variables for applying microaggregation only within strata defined by the variables
- **...**: see arguments below
- **method**: pca, rmd, onedims, single, simple, clustpca, pppca, clustpppca, mdav, clustmcda-pca, influence, mcdpca
number of cluster, if the chosen method performs cluster analysis
sampling weights. If determined, a weighted version of the aggregation measure is chosen automatically, e.g. weighted median or weighted mean.
cluster method, if necessary
 experimental
aggregation statistic, mean, median, trim, onestep (default = mean)
trimming percentage, if measure=trim
variable for sorting, if method= single
transformation for data x

On 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 ‘onedims’, however, often this method is named ‘individual ranking’), but multivariate statistics are strong affected.
With method ‘simple’ one can apply microaggregation directly on the (unsorted) data. It is useful for the comparison with other methods as a benchmark, i.e. replies the question how much better is a sorting of the data before aggregation.
Value

- **mx**: aggregated data set
- **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](http://www.springerlink.com/content/v257655u88w2/?sortorder=asc&p_o=20)


- [http://www.tdp.cat/issues/abs.a/zero.noslash/zero.noslash4a/zero.noslash8.php](http://www.tdp.cat/issues/abs.a/zero.noslash/zero.noslash4a/zero.noslash8.php)


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

Description

Small artificial toy data set.

Usage

```r
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

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

Description

Barplot for objects from class localSuppression.

Usage

```r
## 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)
l1 <- localSuppression(francdat, keyVars=c(2,4,5,6))
l1
plot(l1)
```

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.
**pram**

**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)
```

---

**pram**

**Post RAndomisation Method (PRAM)**

**Description**
PRAM is a probabilistic, perturbative method which can be applied on categorical variables.

**Usage**
```
pram(obj, keyVar=NULL,...) #x, pd=0.8, alpha=0.5)
```

**Arguments**
- **obj**: a numeric vector or factor
- **keyVar**: name of the keyVar that should be prammed
- **...**: see arguments below
- **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**
The method is implemented exactly as described in the citation in the references. First a transition matrix is created in that way, that the diagonal entries of a matrix P are random numbers between ‘pd’ and 1. The remaining entries of the matrix are generated such that the rowSums of the matrix is 1. Then a invariant transition matrix is generated.
Value

- `x` original vector
- `xpramed` the perturbed vector
- `pd` randomly generated diagonal entry of the P (between original pd and 1)
- `Rs` invariant transition matrix
- `alpha` amount of perturbation for the invariant Pram method

Methods

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

Author(s)

Matthias Templ

References

Shlomo, Natalie and de Waal, Ton (2006) Protection of Micro-data Subject to Edit Constraints Against Statistical Disclosure. Southampton, UK, Southampton Statistical Sciences Research Institute, 36pp. (S3RI Methodology Working Papers, M06/16)

Examples

```r
set.seed(123)
x <- factor(sample(1:4, 250, replace=TRUE))
prm1 <- pram(x)
length(which(prm1$x == x))
x2 <- as.factor(sample(1:4, 250, replace=TRUE))
length(which(pram(x2)$x == x2))

data(free1)
free1[,"MARSTAT"] <- as.factor(free1[,"MARSTAT"])
marstatPramed <- pram(free1[,"MARSTAT"])

## 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 <- pram(sdc, keyVar="water")
```
**pram_strata**

---

**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**

```r
pram_strata(obj, variables,...)#, strata_variables = NULL, weights = NULL,
    #seed = NULL, missing = -999, pd = 0.8, alpha = 0.5)
## S3 method for class 'pram_strata'
print(x, ...)
```

**Arguments**

- `obj`: Input data
- `variables`: Names of variables for Post Randomization
- `...`: see arguments below
- `strata_variables`: Names of variables for stratification
- `weights`: a weight for each variable
- `seed`: Integer value for the random seed
- `missing`: A integer value to be used as missing value in the C++ routine
- `x`: Output of pram_strata
- `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**

For each randomized variable a new variable with the suffix ",_pram" will be created.

**Methods**

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

**Note**

The GNU Linear Programming Kit(GLPK) is used in this program. The modified make files from the R package Rglpk are used.
Author(s)

Alexander Kowarik, Bernd Prantner, IHSN C++ source

References

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

See Also

pram

Examples

data(testdata)
res <- pram_strata(testdata, variables="roof", strata.variables=c("urbrur","sex"))
print.pram_strata(res)
res1 <- pram_strata(testdata, variables=c("roof","walls","water"), strata.variables=c("urbrur","sex"))
print.pram_strata(res1)
res2 <- pram_strata(testdata, variables=c("roof","walls","water"), strata.variables=NULL)
print.pram_strata(res2)

## 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 <- pram_strata(sdc, variables=c("urbrur"))

print.freqCalc                   Print method for objects from class freqCalc

Description

Print method for objects from class freqCalc.

Usage

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

Arguments

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

Information about the frequency counts for key variables for object of class ‘freqCalc’.

Author(s)

Matthias Templ

See Also

freqCalc

Examples

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

datatables.frame
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

```r
## 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
```
**print.micro**

*Print method for objects from class micro*

---

**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
```

---

**print.pram**

*Print method for objects from class pram*

---

**Description**

Print method for objects from class 'pram'.

**Usage**

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

x  object from class ‘pram’
   ...
   Additional arguments passed through.

Value

Short information about the method and the parameters used.

Author(s)

Matthias Templ

References


See Also

pram

Examples

data(free1)
x <- free1[,"MARSTAT"]
x2 <- pram(x)
x2

print.suda2  Print method for objects from class suda2

Description

Print method for objects from class suda2.

Usage

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

Arguments

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

Value
Table of dis suda scores.

Author(s)
Matthias Templ

See Also
suda2

Examples

data(testdata)
data_suda2 <- suda2(testdata,variables=c("urbrur","roof","walls","water","sex"))
data_suda2

---

rankSwap

Rank Swapping

Description
Each ranked value is then swapped with another ranked value that has been chosen randomly within a restricted range.

Usage

rankSwap(obj,...)#, variables, TopPercent = 5, BottomPercent = 5, K0 = -1, R0 = 0.95, P = 0, missing = -999, seed = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>obj</td>
<td>matrix or data frame</td>
</tr>
<tr>
<td>...</td>
<td>see arguments below</td>
</tr>
<tr>
<td>variables</td>
<td>names or index of variables for that rank swapping is applied.</td>
</tr>
<tr>
<td>TopPercent</td>
<td>Percentage of largest values that are group together before rank swapping is applied.</td>
</tr>
<tr>
<td>BottomPercent</td>
<td>Percentage of lowest values that are group together before rank swapping is applied.</td>
</tr>
<tr>
<td>K0</td>
<td>Subset-mean preservation factor. Preserves the means before and after rank swapping within a range based on K0.</td>
</tr>
<tr>
<td>R0</td>
<td>Multivariate preservation factor. Preserves the correlation between variables within a certain range based on the given constant R0.</td>
</tr>
<tr>
<td>P</td>
<td>Rank range as percentage of total sample size.</td>
</tr>
<tr>
<td>missing</td>
<td>missing value code.</td>
</tr>
<tr>
<td>seed</td>
<td>Seed.</td>
</tr>
</tbody>
</table>
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.

Value

The rank-swapped data set.

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)

renameVars

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

Description

Change the labels of levels.
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

Examples

## for objects of class sdcMicro:
data(testdata2)
sdc <- createSdcObj(testdata2, keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'), numVars=c('expend','income','savings'), w='sampling_weight')
sdc <- renameVars(sdc, var="urbrur", before=2, after=78)

report Generate a HTML output from an sdcMicroObj

Description
Summary statistics of the original and the perturbed data set

Usage
report(obj, outdir=getwd(),filename="SDC-Report",Title="SDC-Report",...)

Arguments

obj an object of class 'sdcMicroObj'
... see possible arguments below
outdir output folder
filename output filename
Title Title for HTML

Details
The application of this function gives you a html report for your sdcMicro object.
**Methods**

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

**Author(s)**

Matthias Templ

**Examples**

```r
data(free1)
x <- data.frame(free1[,c(2,4:5)])
x["SEX"] <- as.factor(x[,"SEX"])
x["MARSTAT"] <- as.factor(x[,"MARSTAT"])
x["KINDPERS"] <- as.factor(x[,"KINDPERS"])
LLmodGlobalRisk(x, inclProb=1/mean(free1[,"WEIGHT"]))
```

```r
## for objects of class sdcMicro:
## Not run:
data(testdata2)
sdc <- createSdcObj(testdata2, keyVars=c('urbrur','roof','walls','water','electcon','relat','sex'),
    numVars=c('expend','income','savings'), w='sampling_weight')
\dontrun{report(sdc)}
## End(Not run)
```

---

**Deprecated Functions in sdcMicro Package**

**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**

```r
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)
```

**Arguments**

- `x` : data frame or matrix
- `keyVars` : column index of key variables
- `w` : column index of sampling weights
- `importance` : weights for each key variable
sdcMicroObj-class

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

createSdcObj(dat, keyVars, numVars = NULL, weightVar = NULL, hhId = NULL,
             strataVar = NULL, sensibleVar=NULL, options = NULL)
undolast(obj)
nextSdcObj(obj)

Arguments

dat A numeric matrix or data frame containing the data.
obj An object of class 'sdcMicroObj'
keyVars sfdf
numVars sfdf
weightVar sfdf
hhId sfdf
strataVar sfdf
sensibleVar sfdf
options sfdf

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"
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"
manipNumVars: Object of class "dataframeOrNULL"
manipStrataVar: Object of class "factorOrNULL"
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"
set: Object of class "listOrNULL"
prev: Object of class "sdcmicroOrNULL"

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, Elias Rut, Alexander Kowarik, Matthias Templ

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 addNoise without Parameters
sdc <- addNoise(sdc,variables=c("expend","income"))
```
```r
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(sdc@risk$individual)
sdc@risk$global
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
sdc <- rankSwap(sdc)
head(get.sdcMicroObj(sdc, type="manipNumVars"))
sdc@risk$global

### suda2
sdc <- suda2(sdc)
sdc@risk$suda2
```
### topBotCoding

```r
define manipNumVars
sdc@risk$numeric
sdc <- topBotCoding(sdc, value=6/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash, replacement=62/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash/zero.noslash, column="income")
define manipNumVars
```

### LocalRecProg

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

### LLmodGlobalRisk

```r
sdc <- undolast(sdc)
sdc <- LLmodGlobalRisk(sdc, inclProb=0.001)
sdc@risk$model
```

## 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 = "MM", gadp = TRUE)
```

**Arguments**

- **obj**: 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).
- **...** see possible arguments below
- **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.
- **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 originial data is returned.
shuffle

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

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

Methods

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”
...        see arguments below
variables  Categorical (key) variables. Either the column names or and index of the vari-
            ables 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.
The contribution of each key variable to the SUDA score, calculated for each row.

The suda score.

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

data(testdata2)
data_suda2 <- suda2(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)
Description

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

Usage

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

Arguments

- **object**: object from class freqCalc
- **...**: Additional arguments passed through.

Details

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

Value

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

Author(s)

Matthias Templ

See Also

freqCalc

Examples

## example from Capobianchi, Polettini and Lucarelli:
data(francdat)
f <- freqCalc(francdat, keyVars=c(2,4,5,6),w=8)
f
f$fk
f$Fk
## individual risk calculation:
indivf <- indivRisk(f)
indivf$rk
## Local Suppression
localS <- localSupp(f, keyVar=2, indivRisk=indivf$rk, threshold=0.25)
f2 <- freqCalc(localS$freqCalc, keyVars=c(4,5,6), w=8)
summary(f2)
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)
- `adits` 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)
m1 <- microaggregation(Tarragona, method="onedims", aggr=3)
### summary(m1)

summary.pram  Summary method for objects from class pram

Description
Summary method for objects from class ‘pram’ to provide information about transitions.

Usage
### S3 method for class ‘pram’
summary(object, ...)

Arguments
object  object from class ‘pram’
...  Additional arguments passed through.

Details
Shows various information about the transitions.

Value
The summary of object from class ‘pram’.

Author(s)
Matthias Templ
swappNum

References

See Also
pram

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

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

---

Description
Rank Swapping.

Usage
```r
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` percente 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
**Tarragona**

**Author(s)**

Matthias Templ

**References**

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

**See Also**

microaggregation

**Examples**

```r
## Numerical Rank Swapping:
data(free1)
free1[, 31:34] <- rankSwap(free1[, 31:34], P=1/zero.noslash)
```

---

**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)

testdata 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>urbrur</td>
<td>numeric vector</td>
</tr>
<tr>
<td>roof</td>
<td>numeric vector</td>
</tr>
<tr>
<td>walls</td>
<td>numeric vector</td>
</tr>
<tr>
<td>water</td>
<td>numeric vector</td>
</tr>
<tr>
<td>electcon</td>
<td>numeric vector</td>
</tr>
<tr>
<td>relat</td>
<td>numeric vector</td>
</tr>
<tr>
<td>sex</td>
<td>numeric vector</td>
</tr>
<tr>
<td>age</td>
<td>numeric vector</td>
</tr>
<tr>
<td>hhcivil</td>
<td>numeric vector</td>
</tr>
<tr>
<td>expend</td>
<td>numeric vector</td>
</tr>
<tr>
<td>income</td>
<td>numeric vector</td>
</tr>
<tr>
<td>savings</td>
<td>numeric vector</td>
</tr>
<tr>
<td>ori_hid</td>
<td>numeric vector</td>
</tr>
</tbody>
</table>
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

data(testdata)
## maybe str(testdata) ; plot(testdata) ...
topBotCoding  

Top and Bottom Coding  

Description
Function for Top and Bottom Coding.

Usage

\texttt{topBotCoding(obj,...)\#x, value, replacement, kind = "top"}

Arguments

\begin{itemize}
\item obj \quad vector or one-dimensional matrix or data.frame
\item \ldots \quad see arguments below
\item value \quad limit, from where it should be top- or bottom-coded
\item replacement \quad replacement value.
\item kind \quad top or bottom
\end{itemize}

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

Value
Top or bottom coded data.

Methods

\begin{verbatim}
signature(obj = "ANY")
signature(obj = "sdcMicroObj")
\end{verbatim}

Author(s)
Matthias Templ

See Also

\texttt{indivRisk}
valTable

Examples

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

```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 <- topBotCoding(sdc, value=60000000, replacement=62000000, column="income")
```

---

**Comparison of different microaggregation methods**

**Description**

A Function for the comparison of different perturbation methods.

**Usage**

```r
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 = /zero.noslash.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**

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

Value

Measures of information loss splitted for 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)
## valTable(Tarragona[[1/zero.noslash/zero.noslash:2/zero.noslash/zero.noslash]], method=c("simple","onedims","pca","addNoise: additive"))
## valTable(Tarragona, method=c("simple","onedims","pca","clustpppca","mdav","addNoise: additive","swappNum"))
## clustpppca in combination with Mclust outperforms the other algorithms for this data set...

---

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

varToNumeric(obj, var)
varToFactor(obj, var)

Arguments

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

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, 41
  - plotMicro, 42
*Topic **classes**
  - sdcMicroObj-class, 55
*Topic **datasets**
  - casc1, 10
  - CASCrefmicrodata, 11
  - EIA, 18
  - francdat, 20
  - free1, 21
  - microData, 41
  - Tarragona, 67
  - testData, 68
*Topic **manip**
  - addNoise, 7
  - dataGen, 12
  - dRisk, 13
  - dRiskRMD, 14
  - dUtility, 16
  - freqCalc, 22
  - globalRecode, 24
  - indivRisk, 26
  - LLmodGlobalRisk, 27
  - LocalRecProg, 28
  - localSupp, 30
  - localSupp2, 31
  - localSupp2Wrapper, 33
  - localSuppression, 34
  - measure_risk, 36
  - microaggregation, 38
  - pram, 43
  - pram_strata, 45
  - renameVars, 52
  - shuffle, 58
  - suda2, 60
  - swappNum, 65
  - swappNum-deprecated, 66
  - topBotCoding, 70

*Topic **methods**
  - calcRisks, 9
  - groupVars, 25
  - report, 53
  - varToFactor, 72
*Topic **package**
  - sdcMicro-package, 3
*Topic **print**
  - print.freqCalc, 46
  - print.indivRisk, 47
  - print.localSuppression, 48
  - print.micro, 49
  - print.pram, 49
  - print.suda2, 50
  - summary.freqCalc, 62
  - summary.micro, 63
  - summary.pram, 64
  - valTable, 71

addNoise, 7
addNoise, data.frame-method (addNoise), 7
addNoise, matrix-method (addNoise), 7
addNoise, sdcMicroObj-method (addNoise), 7
addNoise-methods (addNoise), 7
calcRisks, 9
calcRisks, sdcMicroObj-method (calcRisks), 9
calcRisks-methods (calcRisks), 9
casc1, 10
CASCrefmicrodata, 11
createSdcObj (sdcMicroObj-class), 55
cut, 25
dataGen, 12
dataGen, data.frame-method (dataGen), 12
dataGen, matrix-method (dataGen), 12
dataGen, sdcMicroObj-method (dataGen), 12
dataGen-methods (dataGen), 12
<table>
<thead>
<tr>
<th>Function</th>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>dRisk</td>
<td>data.frame-method</td>
<td>13</td>
</tr>
<tr>
<td>dRisk</td>
<td>matrix-method</td>
<td>13</td>
</tr>
<tr>
<td>dRisk</td>
<td>sdcMicroObj-method</td>
<td>13</td>
</tr>
<tr>
<td>dRisk</td>
<td>methods</td>
<td>13</td>
</tr>
<tr>
<td>dRiskRMD</td>
<td>data.frame-method</td>
<td>14</td>
</tr>
<tr>
<td>dRiskRMD</td>
<td>matrix-method</td>
<td>14</td>
</tr>
<tr>
<td>dRiskRMD</td>
<td>sdcMicroObj-method</td>
<td>14</td>
</tr>
<tr>
<td>dRiskRMD</td>
<td>methods</td>
<td>14</td>
</tr>
<tr>
<td>dUtility</td>
<td>data.frame-method</td>
<td>16</td>
</tr>
<tr>
<td>dUtility</td>
<td>matrix-method</td>
<td>16</td>
</tr>
<tr>
<td>dUtility</td>
<td>sdcMicroObj-method</td>
<td>16</td>
</tr>
<tr>
<td>dUtility</td>
<td>methods</td>
<td>16</td>
</tr>
<tr>
<td>EIA</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>francdat</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>free1</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>freqCalc</td>
<td></td>
<td>22, 27, 31, 33, 34, 37, 47, 62</td>
</tr>
<tr>
<td>get.sdcMicroObj</td>
<td>sdcMicroObj-class</td>
<td>55</td>
</tr>
<tr>
<td>get.sdcMicroObj</td>
<td>sdcMicroObj-method</td>
<td>55</td>
</tr>
<tr>
<td>globalRecode</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>globalRecode</td>
<td>ANY-method</td>
<td>24</td>
</tr>
<tr>
<td>globalRecode</td>
<td>sdcMicroObj-method</td>
<td>24</td>
</tr>
<tr>
<td>globalRecode</td>
<td>methods</td>
<td>24</td>
</tr>
<tr>
<td>groupVars</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>groupVars</td>
<td>sdcMicroObj-method</td>
<td>25</td>
</tr>
<tr>
<td>groupVars</td>
<td>methods</td>
<td>25</td>
</tr>
<tr>
<td>indivRisk</td>
<td></td>
<td>23, 26, 31, 37, 47, 70</td>
</tr>
<tr>
<td>lddiversity</td>
<td>(measure_risk)</td>
<td>36</td>
</tr>
<tr>
<td>LLmodGlobalRisk</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>LLmodGlobalRisk</td>
<td>ANY-method</td>
<td>27</td>
</tr>
<tr>
<td>LLmodGlobalRisk</td>
<td>data.frame-method</td>
<td>27</td>
</tr>
<tr>
<td>LLmodGlobalRisk</td>
<td>matrix-method</td>
<td>27</td>
</tr>
<tr>
<td>LLmodGlobalRisk</td>
<td>sdcMicroObj-method</td>
<td>27</td>
</tr>
<tr>
<td>LLmodGlobalRisk</td>
<td>methods</td>
<td>27</td>
</tr>
<tr>
<td>lm</td>
<td></td>
<td>39</td>
</tr>
<tr>
<td>LocalRecProg</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>LocalRecProg</td>
<td>data.frame-method</td>
<td>28</td>
</tr>
<tr>
<td>LocalRecProg</td>
<td>matrix-method</td>
<td>28</td>
</tr>
<tr>
<td>LocalRecProg</td>
<td>methods</td>
<td>28</td>
</tr>
<tr>
<td>LocalSupp</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>LocalSupp</td>
<td>ANY-method</td>
<td>30</td>
</tr>
<tr>
<td>LocalSupp</td>
<td>sdcMicroObj-method</td>
<td>30</td>
</tr>
<tr>
<td>LocalSupp</td>
<td>methods</td>
<td>30</td>
</tr>
<tr>
<td>LocalSupp2</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>localSupp2Wrapper</td>
<td></td>
<td>33</td>
</tr>
<tr>
<td>localSuppression</td>
<td></td>
<td>33, 34, 42, 48, 55</td>
</tr>
<tr>
<td>localSuppression</td>
<td>data.frame-method</td>
<td>34</td>
</tr>
<tr>
<td>localSuppression</td>
<td>matrix-method</td>
<td>34</td>
</tr>
<tr>
<td>localSuppression</td>
<td>sdcMicroObj-method</td>
<td>34</td>
</tr>
<tr>
<td>localSuppression</td>
<td>methods</td>
<td>34</td>
</tr>
<tr>
<td>loglm</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>measure_risk</td>
<td></td>
<td>23, 28, 36</td>
</tr>
<tr>
<td>measure_risk</td>
<td>data.frame-method</td>
<td>36</td>
</tr>
<tr>
<td>measure_risk</td>
<td>matrix-method</td>
<td>36</td>
</tr>
<tr>
<td>measure_risk</td>
<td>sdcMicroObj-method</td>
<td>36</td>
</tr>
<tr>
<td>measure_risk</td>
<td>methods</td>
<td>36</td>
</tr>
<tr>
<td>microaggregation</td>
<td></td>
<td>38, 43, 49, 64, 66, 67, 72</td>
</tr>
<tr>
<td>microaggregation</td>
<td>ANY-method</td>
<td>38</td>
</tr>
<tr>
<td>microaggregation</td>
<td>sdcMicroObj-method</td>
<td>38</td>
</tr>
<tr>
<td>microaggregation</td>
<td>methods</td>
<td>38</td>
</tr>
</tbody>
</table>
microaggregation-methods
  (microaggregation), 38
microData, 41
nextSdcObj(sdcMicroObj-class), 55
nextSdcObj, sdcMicroObj-method(sdcMicroObj-class), 55
plot.localSuppression, 41
plotMicro, 40, 42
pram, 43, 50, 65
pram, ANY-method (pram), 43
pram, sdcMicroObj-method (pram), 43
pram-methods (pram), 43
pram_strata, 45
pram_strata, data.frame-method (pram_strata), 45
pram_strata, matrix-method (pram_strata), 45
pram_strata, sdcMicroObj-method (pram_strata), 45
pram_strata-methods (pram_strata), 45
print.freqCalc, 46
print.indivRisk, 47
print.ldiversity (measure_risk), 36
print.localSuppression, 48
print.measure_risk (measure_risk), 36
print.micro, 49
print.pram, 49
print.pram_strata (pram_strata), 45
print.suda2, 50
rankSwap, 51, 59
rankSwap, data.frame-method (rankSwap), 51
rankSwap, matrix-method (rankSwap), 51
rankSwap, sdcMicroObj-method (rankSwap), 51
rankSwap-methods (rankSwap), 51
renameVars, 52
renameVars, sdcMicroObj-method (renameVars), 52
renameVars-methods (renameVars), 52
report, 53
report, sdcMicroObj-method (report), 53
report-methods (report), 53
sdcMicro (sdcMicro-package), 3
sdcMicro-deprecated, 54
sdcMicro-package, 3
sdcMicroObj-class, 55
set.sdcMicroObj (sdcMicroObj-class), 55
set.sdcMicroObj, sdcMicroObj, character, listOrNULL-method
  (sdcMicroObj-class), 55
shuffle, 58
shuffle, data.frame-method (shuffle), 58
shuffle, matrix-method (shuffle), 58
shuffle, sdcMicroObj-method (shuffle), 58
shuffle-methods (shuffle), 58
suda2, 51, 60
suda2, data.frame-method (suda2), 60
suda2, matrix-method (suda2), 60
suda2, sdcMicroObj-method (suda2), 60
suda2-methods (suda2), 60
summary.freqCalc, 62
summary.micro, 9, 40, 63, 72
summary.pram, 64
swappNum, 65
swappNum-deprecated, 66
Tarragona, 67
testdata, 68
testdata2 (testdata), 68
topBotCoding, 70
topBotCoding, ANY-method (topBotCoding), 70
topBotCoding, sdcMicroObj-method (topBotCoding), 70
topBotCoding-methods (topBotCoding), 70
undolast (sdcMicroObj-class), 55
undolast, sdcMicroObj-method (sdcMicroObj-class), 55
valTable, 40, 64, 71
varToFactor, 72
varToFactor, sdcMicroObj-method (varToFactor), 72
varToFactor-methods (varToFactor), 72
varToNumeric (varToFactor), 72
varToNumeric, sdcMicroObj-method (varToFactor), 72
varToNumeric-methods (varToFactor), 72