Package 'SmallCountRounding'

November 17, 2026

Type Package

Title Small Count Rounding of Tabular Data

Version 1.2.7

Date 2026-11-17

Author Øyvind Langsrud [aut, cre],
Johan Heldal [aut]

Maintainer Øyvind Langsrud <oyl@ssb.no>

Imports methods, Matrix, SSBtools (>= 1.8.2), ellipsis, rlang

VignetteBuilder knitr

Suggests knitr, rmarkdown, kableExtra, sdcHierarchies, testthat,
data.table

Description A statistical disclosure control tool to protect frequency tables in cases where small values are sensitive. The function PLSrounding() performs small count rounding of necessary inner cells so that all small frequencies of cross-classifications to be published (publishable cells) are rounded. This is equivalent to changing micro data since frequencies of unique combinations are changed. Thus, additivity and consistency are guaranteed. The methodology is described in Langsrud and Heldal (2018) <https://www.researchgate.net/publication/327768398_An_Algorithm_for_Small_Count_Rounding_of_Tabular_Data>.

License MIT + file LICENSE

URL https://github.com/statisticsnorway/ssb-smallcountrounding,
https://statisticsnorway.github.io/ssb-smallcountrounding/

BugReports https://github.com/statisticsnorway/ssb-smallcountrounding/issues

Encoding UTF-8

Config/roxygen2/version 8.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-11-17 09:10:03 UTC

Contents

SmallCountRounding-package 2
FormulaSelection.PLSrounded 3
HD 3
PLS2way 4
PLSrounding 5
PLSroundingFits 10
PLSroundingLoop 13
print.PLSrounded 14
RoundViaDummy 15
SmallCountData 19

Index **20**

`SmallCountRounding-package`

Small Count Rounding of Tabular Data

Description

A statistical disclosure control tool to protect frequency tables in cases where small values are sensitive. The main function, `PLSrounding`, performs small count rounding of necessary inner cells (Heldal, 2017) so that all small frequencies of cross-classifications to be published (publishable cells) are rounded. This is equivalent to changing micro data since frequencies of unique combinations are changed. Thus, additivity and consistency are guaranteed. This is performed by an algorithm inspired by partial least squares regression (Langsrud and Heldal, 2018).

Author(s)

Maintainer: Øyvind Langsrud <oyl@ssb.no>

Authors:

- Øyvind Langsrud <oyl@ssb.no>
- Johan Heldal

References

Heldal, J. (2017): "The European Census Hub 2011 Hypercubes - Norwegian SDC Experiences". In: *Work Session on Statistical Data Confidentiality*, Skopje, The former Yugoslav Republic of Macedonia, September 20-22 , 2017.

Langsrud, Ø. and Heldal, J. (2018): "An Algorithm for Small Count Rounding of Tabular Data". Presented at: *Privacy in statistical databases*, Valencia, Spain. September 26-28, 2018. https://www.researchgate.net/publication/327768398_An_Algorithm_for_Small_Count_Rounding_of_Tabular_Data

See Also

Useful links:

- https://github.com/statisticsnorway/ssb-smallcountrounding
- https://statisticsnorway.github.io/ssb-smallcountrounding/
- Report bugs at https://github.com/statisticsnorway/ssb-smallcountrounding/issues

---

`FormulaSelection.PLSrounded` *FormulaSelection method for PLSrounded*

Description

FormulaSelection method for PLSrounded

Usage

```
## S3 method for class 'PLSrounded'
FormulaSelection(x, formula = NULL, intercept = NA, logical = FALSE)
```

Arguments

`x`	PLSrounded object
`formula`	`formula` parameter to `FormulaSelection`. When `NULL` (default), the publish data frame is returned without any limitation.
`intercept`	`intercept` parameter to `FormulaSelection`.
`logical`	`logical` parameter to `FormulaSelection`.

Value

Limited version of the publish data frame

`HD` *Hellinger Distance (Utility)*

Description

Hellinger distance (`HD`) and a related utility measure (`HDutility`) described in the reference below. The utility measure is made to be bounded between 0 and 1.

Usage

```
HD(f, g)

HDutility(f, g)
```

Arguments

f	Vector of original counts
g	Vector of perturbed counts

Details

HD is defined as "`sqrt(sum((sqrt(f) - sqrt(g))^2)/2)`" and HDutility is defined as "`1 - HD(f, g)/sqrt(sum(f))`".

Value

Hellinger distance or related utility measure

References

Shlomo, N., Antal, L., & Elliot, M. (2015). Measuring Disclosure Risk and Data Utility for Flexible Table Generators, Journal of Official Statistics, 31(2), 305-324. doi:10.1515/jos20150019

Examples

```
f <- 1:6
g <- c(0, 3, 3, 3, 6, 6)
print(c(
  HD = HD(f, g),
  HDutility = HDutility(f, g),
  maxdiff = max(abs(g - f)),
  meanAbsDiff = mean(abs(g - f)),
  rootMeanSquare = sqrt(mean((g - f)^2))
))
```

PLS2way *Two-way table from PLSrounding output*

Description

Output from PLSrounding is presented as two-way table(s) in cases where this is possible. A requirement is that the number of main dimensional variables is two.

Usage

```
PLS2way(obj, variable = c("rounded", "original", "difference", "code"))
```

Arguments

obj	Output object from PLSrounding
variable	One of "rounded" (default), "original", "difference" or "code".

Details

When parameter "variable" is "code", output is coded as "#" (publish), "." (inner) and "&" (both).

Value

A data frame

Examples

```
# Making tables from PLSrounding examples
z <- SmallCountData("e6")
a <- PLSrounding(z, "freq", formula = ~eu * year + geo)
PLS2way(a, "original")
PLS2way(a, "difference")
PLS2way(a, "code")
PLS2way(PLSrounding(z, "freq", formula = ~eu * year + geo * year), "code")
eHrc2 <- list(geo = c("EU", "@Portugal", "@Spain", "Iceland"), year = c("2018", "2019"))
PLS2way(PLSrounding(z, "freq", hierarchies = eHrc2))
```

PLSrounding *PLS inspired rounding*

Description

Small count rounding of necessary inner cells are performed so that all small frequencies of cross-classifications to be published (publishable cells) are rounded. The publishable cells can be defined from a model formula, hierarchies or automatically from data.

Usage

```
PLSrounding(
  data,
  freqVar = NULL,
  roundBase = 3,
  hierarchies = NULL,
  formula = NULL,
  dimVar = NULL,
  maxRound = roundBase - 1,
  printInc = nrow(data) > 1000,
  output = NULL,
  extend0 = FALSE,
  preAggregate = NA,
  zeroCandidates = FALSE,
  forceInner = FALSE,
  allSmall = FALSE,
  step = 0,
  aggregatePackage = "base",
```

```
  aggregateNA = TRUE,
  aggregateBaseOrder = FALSE,
  rowGroupsPackage = aggregatePackage,
  ...,
 action_unused_dots = getOption("SmallCountRounding.action_unused_dots", "warn"),
 allowed_unused_dots = getOption("SmallCountRounding.allowed_unused_dots", character(0))
)

PLSroundingInner(..., output = "inner")

PLSroundingPublish(..., output = "publish")
```

Arguments

data	Input data (inner cells), typically a data frame, tibble, or data.table. If data is not a classic data frame, it will be coerced to one internally unless preAggregate is TRUE and aggregatePackage is "data.table".
freqVar	Variable holding counts (inner cells frequencies). When NULL (default), microdata is assumed.
roundBase	Rounding base
hierarchies	List of hierarchies
formula	Model formula defining publishable cells
dimVar	The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified.
maxRound	Inner cells contributing to original publishable cells equal to or less than maxRound will be rounded
printInc	Printing iteration information to console when TRUE
output	Possible non-NULL values are "input", "inner" and "publish". Then a single data frame is returned.
extend0	When extend0 is set to TRUE, the data is automatically extended. This is relevant when zeroCandidates = TRUE (see RoundViaDummy). Additionally, extend0 can be specified as a list, representing the varGroups parameter in the Extend0 function. Can also be set to "all" which means that input codes in hierarchies are considered in addition to those in data.
preAggregate	When TRUE, the data will be aggregated within the function to an appropriate level. This is defined by the dimensional variables according to dimVar, hierarchies or formula. When FALSE, no aggregation is performed. When NA (default), the function will automatically decide whether to aggregate: aggregation is applied unless freqVar is present and the data contain no duplicated rows with respect to the dimensional variables. Exception: if a non-NULL x (the model matrix) is supplied via ..., NA is treated as FALSE.
zeroCandidates	When TRUE, inner cells with zero frequencies are treated as candidates to be rounded up. See RoundViaDummy() for details. Note: When using microdata input (freqVar = NULL), extend0 must be specified for zeroCandidates to have any effect.

forceInner
: When TRUE, all inner cells are forced to be rounded. See RoundViaDummy() for details.

allSmall
: When TRUE, all small inner cells (<= maxRound) are rounded. See RoundViaDummy() for details.

step
: A parameter controlling forward and backward steps within the algorithm. The step parameter can significantly impact performance on large datasets. For example, using step = list(100) may be a useful approach. See the parameter description and details in the underlying function RoundViaDummy().

aggregatePackage
: Package used to preAggregate. Parameter pkg to aggregate_by_pkg.

aggregateNA
: Whether to include NAs in the grouping variables while preAggregate. Parameter include_na to aggregate_by_pkg.

aggregateBaseOrder
: Parameter base_order to aggregate_by_pkg, used when preAggregate. The default is set to FALSE to avoid unnecessary sorting operations. When TRUE, an attempt is made to return the same result with data.table as with base R. This cannot be guaranteed due to potential variations in sorting behavior across different systems.

rowGroupsPackage
: Parameter pkg to RowGroups. The parameter is input to Formula2ModelMatrix via ModelMatrix.

...
: Further parameters sent to RoundViaDummy

action_unused_dots
: Character string controlling how unused arguments in ... are handled. Internally uses ellipsis::check_dots_used() with a custom action. One of "warn", "abort", "inform", or "none". The value "none" disables the check entirely. The default is taken from getOption("SmallCountRounding.action_unused_dots"), falling back to "warn" if the option is not set. Users can change the default globally with e.g. options(SmallCountRounding.action_unused_dots = "abort").

allowed_unused_dots
: Character vector of argument names ignored by the unused-argument check. May be useful when this function is wrapped by another function, or in other cases where a correctly spelled argument is nevertheless not registered as used. The default is taken from getOption("SmallCountRounding.allowed_unused_dots"), falling back to character(0) if the option is not set. Users can change the default globally with e.g. options(SmallCountRounding.allowed_unused_dots = c("plotColor", "lineType")).

Details

This function is a user-friendly wrapper for RoundViaDummy with data frame output and with computed summary of the results. See RoundViaDummy for more details.

Value

Output is a four-element list with class attribute "PLSrounded", which ensures informative printing and enables the use of FormulaSelection on this object.

inner	Data frame corresponding to input data with the main dimensional variables and with cell frequencies (original, rounded, difference).
publish	Data frame of publishable data with the main dimensional variables and with cell frequencies (original, rounded, difference).
metrics	A named character vector of various statistics calculated from the two output data frames ("inner_" used to distinguish). See examples below and the function HDutility.
freqTable	Matrix of frequencies of cell frequencies and absolute differences. For example, row "rounded" and column "inn.4+" is the number of rounded inner cell frequencies greater than or equal to 4.

References

Langsrud, Ø. and Heldal, J. (2018): "An Algorithm for Small Count Rounding of Tabular Data". Presented at: *Privacy in statistical databases*, Valencia, Spain. September 26-28, 2018. https://www.researchgate.net/publication/327768398_An_Algorithm_for_Small_Count_Rounding_of_Tabular_Data

See Also

RoundViaDummy, PLS2way, ModelMatrix

Examples

```
# Small example data set
z <- SmallCountData("e6")
print(z)

# Publishable cells by formula interface
a <- PLSrounding(z, "freq", roundBase = 5,  formula = ~geo + eu + year)
print(a)
print(a$inner)
print(a$publish)
print(a$metrics)
print(a$freqTable)

# Using FormulaSelection()
FormulaSelection(a$publish, ~eu + year)
FormulaSelection(a, ~eu + year) # same as above
FormulaSelection(a)             # just a$publish

# Recalculation of maxdiff, HDutility, meanAbsDiff and rootMeanSquare
max(abs(a$publish[, "difference"]))
HDutility(a$publish[, "original"], a$publish[, "rounded"])
mean(abs(a$publish[, "difference"]))
sqrt(mean((a$publish[, "difference"])^2))

# Five lines below produce equivalent results
# Ordering of rows can be different
PLSrounding(z, "freq", dimVar = c("geo", "eu", "year"))
```

```
PLSrounding(z, "freq", formula = ~eu * year + geo * year)
PLSrounding(z[, -2], "freq", hierarchies = SmallCountData("eHrc"))
PLSrounding(z[, -2], "freq", hierarchies = SmallCountData("eDimList"))
PLSrounding(z[, -2], "freq", hierarchies = SmallCountData("eDimList"), formula = ~geo * year)

# Define publishable cells differently by making use of formula interface
PLSrounding(z, "freq", formula = ~eu * year + geo)

# Define publishable cells differently by making use of hierarchy interface
eHrc2 <- list(geo = c("EU", "@Portugal", "@Spain", "Iceland"), year = c("2018", "2019"))
PLSrounding(z, "freq", hierarchies = eHrc2)

# Also possible to combine hierarchies and formula
PLSrounding(z, "freq", hierarchies = SmallCountData("eDimList"), formula = ~geo + year)

# Single data frame output and total codes
PLSroundingInner(z, "freq", roundBase = 5, formula = ~geo + eu + year)
PLSroundingPublish(z, roundBase = 5, formula = ~geo + eu + year)
PLSroundingPublish(z, roundBase = 5, formula = ~geo + eu + year,
                   total = c(geo = "Europe", year = "All_years"))

# Microdata input
PLSroundingInner(rbind(z, z), roundBase = 5, formula = ~geo + eu + year)

# Zero perturbed due to both  extend0 = TRUE and zeroCandidates = TRUE
set.seed(12345)
PLSroundingInner(z[sample.int(5, 12, replace = TRUE), 1:3],
                 formula = ~geo + eu + year, roundBase = 5,
                 extend0 = TRUE, zeroCandidates = TRUE, printInc = TRUE)

# Parameter avoidHierarchical (see RoundViaDummy and ModelMatrix)
PLSroundingPublish(z, roundBase = 5, formula = ~geo + eu + year, avoidHierarchical = TRUE)


# To illustrate hierarchical_extend0
#    (parameter to underlying function, SSBtools::Extend0fromModelMatrixInput)
PLSroundingInner(z[-c(2:3), ], roundBase = 5, formula = ~geo + eu + year,
   avoidHierarchical = TRUE, zeroCandidates = TRUE, extend0 = TRUE)
PLSroundingInner(z[-c(2:3), ], roundBase = 5, formula = ~geo + eu + year,
   avoidHierarchical = TRUE, zeroCandidates = TRUE, extend0 = TRUE,
   hierarchical_extend0 = TRUE)

# Package sdcHierarchies can be used to create hierarchies.
# The small example code below works if this package is available.
if (require(sdcHierarchies)) {
  z2 <- cbind(geo = c("11", "21", "22"), z[, 3:4], stringsAsFactors = FALSE)
  h2 <- list(
    geo = hier_compute(inp = unique(z2$geo), dim_spec = c(1, 1), root = "Tot", as = "df"),
    year = hier_convert(hier_create(root = "Total", nodes = c("2018", "2019")), as = "df"))
  PLSrounding(z2, "freq", hierarchies = h2)
}

# Use PLS2way to produce tables as in Langsrud and Heldal (2018) and to demonstrate
```

```
# parameters maxRound, zeroCandidates and identifyNew (see RoundViaDummy).
# Parameter rndSeed used to ensure same output as in reference.
exPSD <- SmallCountData("exPSD")
a <- PLSrounding(exPSD, "freq", 5, formula = ~rows + cols, rndSeed=124)
PLS2way(a, "original")  # Table 1
PLS2way(a)  # Table 2
a <- PLSrounding(exPSD, "freq", 5, formula = ~rows + cols, identifyNew = FALSE, rndSeed=124)
PLS2way(a)  # Table 3
a <- PLSrounding(exPSD, "freq", 5, formula = ~rows + cols, maxRound = 7)
PLS2way(a)  # Values in col1 rounded
a <- PLSrounding(exPSD, "freq", 5, formula = ~rows + cols, zeroCandidates = TRUE)
PLS2way(a)  # (row3, col4): original is 0 and rounded is 5

# Using formula followed by FormulaSelection
output <- PLSrounding(data = SmallCountData("example1"),
                      formula = ~age * geo * year + eu * year,
                      freqVar = "freq",
                      roundBase = 5)
FormulaSelection(output, ~(age + eu) * year)

# Example similar to the one in the documentation of tables_by_formulas,
# but using PLSroundingPublish with roundBase = 4.
tables_by_formulas(SSBtoolsData("magnitude1"),
                   table_fun = PLSroundingPublish,
                   table_formulas = list(table_1 = ~region * sector2,
                                         table_2 = ~region1:sector4 - 1,
                                         table_3 = ~region + sector4 - 1),
                   substitute_vars = list(region = c("geo", "eu"), region1 = "eu"),
                   collapse_vars = list(sector = c("sector2", "sector4")),
                   roundBase = 4)
```

PLSroundingFits *Small count rounding with post-processing to expected frequencies*

Description

The counts rounded by PLSrounding Thereafter, based on the publishable rounded data, expected inner cell frequencies are generated by iterative proportional fitting using Mipf. To ensure that empty cells missing in input data are included in the fitting process, the data is first extended using Extend0.

Usage

```
PLSroundingFits(
  data,
  freqVar = NULL,
  roundBase = 3,
  hierarchies = NULL,
```

```
  formula = NULL,
  dimVar = NULL,
  preAggregate = is.null(freqVar),
  printInc = nrow(data) > 1000,
  xReturn = FALSE,
  extend0 = FALSE,
  extend0Fits = TRUE,
  limit = 1e-10,
  viaQR = FALSE,
  iter = 1000,
  eps = 0.01,
  tol = 1e-13,
  reduceBy0 = TRUE,
  reduceByColSums = TRUE,
  reduceByLeverage = FALSE,
  ...
)
```

Arguments

data	data frame (inner cells)
freqVar	Variable holding counts
roundBase	Rounding base
hierarchies	List of hierarchies
formula	Model formula
dimVar	Dimensional variables
preAggregate	Aggregation
printInc	Printing iteration information
xReturn	Dummy matrix in output when TRUE. To return crossTable as well, use xReturn = 2.
extend0	PLSrounding parameter. See below.
extend0Fits	When extend0Fits is set to TRUE (default), the data is automatically extended. Additionally, extend0Fits can be specified as a list, or set to "all" (see PLSrounding). Previously, this functionality was controlled by a parameter called extend0, but now extend0 is specific to the PLSrounding function. When both extend0 and extend0Fits are used simultaneously, extend0Fits adds an additional extension on top of the one provided by extend0 (see example).
limit	LSfitNonNeg parameter
viaQR	LSfitNonNeg parameter
iter	Mipf parameter
eps	Mipf parameter
tol	Mipf parameter
reduceBy0	Mipf parameter

reduceByColSums
: Mipf parameter

reduceByLeverage
: Mipf parameter

... : Further parameters to PLSrounding.

Details

The nine first parameters is documented in more detail in PLSrounding. If iterative proportional fitting succeeds, the maximum difference between rounded counts and ipFit is less than input parameter eps.

Value

Output from PLSrounding (class attribute "PLSrounded") with modified versions of inner and publish:

inner : Extended with more input data variables and with expected frequencies (ipFit).

publish : Extended with aggregated expected frequencies (ipFit).

Examples

```
z <- data.frame(geo  = c("Iceland", "Portugal", "Spain"),
                eu = c("nonEU", "EU", "EU"),
                year = rep(c("2018","2019"), each = 3),
                freq = c(2,3,7,1,5,6), stringsAsFactors = FALSE)
z4 <- z[-c(1:2), ]

PLSroundingFits(z4, "freq", formula = ~eu * year + geo, extend0 = FALSE)[c("inner", "publish")]
PLSroundingFits(z4, "freq", formula = ~eu * year + geo)[c("inner", "publish")]

my_km2 <- SSBtools::SSBtoolsData("my_km2")

# Default automatic extension (extend0Fits = TRUE)
PLSroundingFits(my_km2, "freq",
     formula = ~(Sex + Age) * Municipality * Square1000m + Square250m)[c("inner", "publish")]

# Manual specification to avoid Nittedal combined with another_km
PLSroundingFits(my_km2, "freq", formula = ~(Sex + Age) * Municipality * Square1000m + Square250m,
       extend0Fits = list(c("Sex", "Age"),
       c("Municipality", "Square1000m", "Square250m")))[c("inner", "publish")]

# Example with both extend0 (specified) and extend0Fits (default is TRUE)
PLSroundingFits(my_km2, "freq", formula = ~(Sex + Age) * Municipality * Square1000m + Square250m,
     printInc = TRUE, zeroCandidates = TRUE, roundBase = 5, extend0 = list(c("Sex", "Age"),
       c("Municipality", "Square1000m", "Square250m")))[c("inner", "publish")]
```

PLSroundingLoop *PLSrounding on portions of data at a time*

Description

The PLSrounding runs are coordinated by using preliminary differences as input for the next run (parameter preDifference)

Usage

```
PLSroundingLoop(
  data,
  loopId,
  ...,
  zeroCandidates = FALSE,
  forceInner = FALSE,
  preRounded = NULL,
  plsWeights = NULL,
  printInc = TRUE,
  preDifference = TRUE,
  preOutput = NULL,
  rndSeed = 123,
  action_unused_dots = "warn"
)
```

Arguments

data	Input data as a data frame (inner cells)
loopId	Variable holding id for loops
...	PLSrounding parameters
zeroCandidates	PLSrounding parameter (see details)
forceInner	PLSrounding parameter (see details)
preRounded	PLSrounding parameter (see details)
plsWeights	PLSrounding parameter (see details)
printInc	Printing iteration information to console when TRUE
preDifference	When TRUE, the preDifference parameter to PLSrounding is used. Each time with the differences obtained so far.
preOutput	preOutput The function can continue from output from a previous run
rndSeed	If non-NULL, a random generator seed to be set locally at the beginning of PLSroundingLoop without affecting the random value stream in R. Within PLSroundingLoop, PLSrounding is called with rndSeed = NULL.
action_unused_dots	PLSrounding parameter.

Details

Note that in this function `zeroCandidates`, `forceInner`, `preRounded` and `plsWeights` cannot be supplied as vectors. They may be specified as functions or as variables in the input data.

Value

As output from `PLSrounding`

Examples

```
mf2 <- ~region + fylke * hovedint
z2 <- SmallCountData("z2")
a <- PLSroundingLoop(z2, loopId = "kostragr", freqVar = "ant", formula = mf2)
a
```

---

print.PLSrounded *Print method for PLSrounded*

Description

Print method for PLSrounded

Usage

```
## S3 method for class 'PLSrounded'
print(x, digits = max(getOption("digits") - 3, 3), ...)
```

Arguments

`x`	PLSrounded object
`digits`	positive integer. Minimum number of significant digits to be used for printing most numbers.
`...`	further arguments sent to the underlying

Value

Invisibly returns the original object.

RoundViaDummy *Small Count Rounding of Tabular Data*

Description

Small count rounding via a dummy matrix and by an algorithm inspired by PLS

Usage

```
RoundViaDummy(
  data,
  freqVar,
  formula = NULL,
  roundBase = 3,
  singleRandom = FALSE,
  crossTable = TRUE,
  total = "Total",
  maxIterRows = 1000,
  maxIter = 1e+07,
  x = NULL,
  hierarchies = NULL,
  xReturn = FALSE,
  maxRound = roundBase - 1,
  zeroCandidates = FALSE,
  forceInner = FALSE,
  identifyNew = TRUE,
  step = 0,
  preRounded = NULL,
  leverageCheck = FALSE,
  easyCheck = TRUE,
  printInc = TRUE,
  rndSeed = 123,
  dimVar = NULL,
  plsWeights = NULL,
  preDifference = NULL,
  allSmall = FALSE,
  ...
)
```

Arguments

data	Input data as a data frame (inner cells)
freqVar	Variable holding counts (name or number)
formula	Model formula defining publishable cells. Will be used to calculate x (via ModelMatrix). When NULL, x must be supplied.
roundBase	Rounding base

singleRandom	Single random draw when TRUE (instead of algorithm)
crossTable	When TRUE, cross table in output and caculations via FormulaSums()
total	String(s) used to name totals, passed to SSBtools::ModelMatrix(). Named vectors or lists are supported.
maxIterRows	See details
maxIter	Maximum number of iterations
x	Dummy matrix defining publishable cells
hierarchies	List of hierarchies, which can be converted by AutoHierarchies. Thus, a single string as hierarchy input is assumed to be a total code. Exceptions are "rowFactor" or "", which correspond to only using the categories in the data.
xReturn	Dummy matrix in output when TRUE (as input parameter x)
maxRound	Inner cells contributing to original publishable cells equal to or less than maxRound will be rounded.
zeroCandidates	When TRUE, inner cells in input with zero count (and multiple of roundBase when maxRound is in use) contributing to publishable cells will be included as candidates to obtain roundBase value. With vector input, the rule is specified individually for each cell. This can be specified as a vector, a variable in data or a function generating it (see details).
forceInner	When TRUE, all inner cells will be rounded. Use vector input to force individual cells to be rounded. This can be specified as a vector, a variable in data or a function generating it (see details). Can be combined with parameter zeroCandidates to allow zeros and roundBase multiples to be rounded up.
identifyNew	When TRUE, new cells may be identified after initial rounding to ensure all rounded publishable cells equal to or less than maxRound to be roundBase multiples. Use NA for the a less conservative behavior (old behavior). Then it is ensured that no nonzero rounded publishable cells are smaller than roundBase. When maxRound is default, there is no difference between TRUE and NA.
step	When step>1, the original forward part of the algorithm is replaced by a kind of stepwise. After step steps forward, backward steps may be performed. The step parameter is also used for backward-forward iteration at the end of the algorithm; step backward steps may be performed. For greater control, the step parameter can be specified as a vector. Additionally, it can be provided as a list to trigger a final re-run iteration. See details.
preRounded	A vector or a variable in data that contains a mixture of missing values and predetermined values of rounded inner cells. Can also be specified as a function generating it (see details).
leverageCheck	When TRUE, all inner cells that depends linearly on the published cells and with small frequencies (<=maxRound) will be rounded. The computation of leverages can be very time and memory consuming. The function Reduce0exact is called. The default leverage limit is 0.999999. Another limit can be sent as input instead of TRUE. Checking is performed before and after (since new zeros) rounding. Extra iterations are performed when needed.
easyCheck	A light version of the above leverage checking. Checking is performed after rounding. Extra iterations are performed when needed. Reduce0exact is called with reduceByLeverage=FALSE and reduceByColSums=TRUE.

printInc	Printing iteration information to console when TRUE
rndSeed	If non-NULL, a random generator seed to be used locally within the function without affecting the random value stream in R.
dimVar	The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified.
plsWeights	A vector of weights for each cell to be published or a function generating it (see details). For use in the algorithm criterion.
preDifference	A data.frame with differences already obtained from rounding another subset of data. There must be columns that match crossTable. Differences must be in the last column.
allSmall	When TRUE, all small inner cells (<= maxRound) are rounded. This parameter is a simplified alternative to specifying forceInner (see details).
...	Further parameters sent to ModelMatrix. In particular, one can specify removeEmpty=TRUE to omit empty combinations. The parameter inputInOutput can be used to specify whether to include codes from input. The parameter avoidHierarchical (Formula2ModelMatrix) can be combined with formula input.

Details

Small count rounding of necessary inner cells are performed so that all small frequencies of cross-classifications to be published (publishable cells) are rounded. This is equivalent to changing micro data since frequencies of unique combinations are changed. Thus, additivity and consistency are guaranteed. The matrix multiplication formula is: `yPublish = t(x) %*% yInner`, where `x` is the dummy matrix.

Parameters `zeroCandidates`, `forceInner`, `preRounded` and `plsWeights` can be specified as functions. The supplied functions take the following arguments: `data`, `yPublish`, `yInner`, `crossTable`, `x`, `roundBase`, `maxRound`, and `...`, where the first two are numeric vectors of original counts. When `allSmall` is `TRUE`, `forceInner` is set to `function(yInner, maxRound, ...) yInner <= maxRound`.

Details about the `step` parameter:

- `step` as a numeric vector is converted to three parameters by
  - `step1 <- step[1]`
  - `step2 <- ifelse(length(step)>=2, step[2], round(step/2))`
  - `step3 <- ifelse(length(step)>=3, step[3], step[1])`

  After `step1` steps forward, up to `step2` backward steps may be performed. At the end of the algorithm; up to `step3` backward steps may be executed repeatedly.
- `step` when provided as a list (of numeric vectors), is adjusted to a length of 3 using `rep_len(step, 3)`.
  - `step[[1]]` is used in the main iterations.
  - `step[[2]]`, when non-`NULL`, is used in a final re-run iteration.
  - `step[[3]]` is used in extra iterations caused by `easyCheck` or `leverageCheck`.

  Setting `step = list(0)` will result in standard behavior, with the exception that an extra re-run iteration is performed. The most detailed setting is achieved by setting `step` to a length-3 list where each element has length 3.

Value

A list where the two first elements are two column matrices. The first matrix consists of inner cells and the second of cells to be published. In each matrix the first and the second column contains, respectively, original and rounded values. By default the cross table is the third element of the output list.

Note

Iterations are needed since after initial rounding of identified cells, new cells are identified. If cases of a high number of identified cells the algorithm can be too memory consuming (unless singleRandom=TRUE). To avoid problems, not more than maxIterRows cells are rounded in each iteration. The iteration limit (maxIter) is by default set to be high since a low number of maxIterRows may need a high number of iterations.

See Also

See the user-friendly wrapper PLSrounding and see Round2 for rounding by other algorithm

Examples

```
# See similar and related examples in PLSrounding documentation
RoundViaDummy(SmallCountData("e6"), "freq")
RoundViaDummy(SmallCountData("e6"), "freq", formula = ~eu * year + geo)
RoundViaDummy(SmallCountData("e6"), "freq", hierarchies =
   list(geo = c("EU", "@Portugal", "@Spain", "Iceland"), year = c("2018", "2019")))

RoundViaDummy(SmallCountData('z2'),
              'ant', ~region + hovedint + fylke*hovedint + kostragr*hovedint, 10)
mf <- ~region*mnd + hovedint*mnd + fylke*hovedint*mnd + kostragr*hovedint*mnd
a <- RoundViaDummy(SmallCountData('z3'), 'ant', mf, 5)
b <- RoundViaDummy(SmallCountData('sosialFiktiv'), 'ant', mf, 4)
print(cor(b[[2]]),digits=12) # Correlation between original and rounded

# Demonstrate parameter leverageCheck
# The 42nd inner cell must be rounded since it can be revealed from the published cells.
mf2 <- ~region + hovedint + fylke * hovedint + kostragr * hovedint
RoundViaDummy(SmallCountData("z2"), "ant", mf2, leverageCheck = FALSE)$yInner[42, ]
RoundViaDummy(SmallCountData("z2"), "ant", mf2, leverageCheck = TRUE)$yInner[42, ]

## Not run:
# Demonstrate parameters maxRound, zeroCandidates and forceInner
# by tabulating the inner cells that have been changed.
z4 <- SmallCountData("sosialFiktiv")
for (forceInner in c("FALSE", "z4$ant < 10"))
  for (zeroCandidates in c(FALSE, TRUE))
    for (maxRound in c(2, 5)) {
      set.seed(123)
      a <- RoundViaDummy(z4, "ant", formula = mf, maxRound = maxRound,
                         zeroCandidates = zeroCandidates,
                         forceInner = eval(parse(text = forceInner)))
      change <- a$yInner[, "original"] != a$yInner[, "rounded"]
```

```
  cat("\n\n--------------------------------------------------\n")
  cat("      maxRound:", maxRound, "\n")
  cat("zeroCandidates:", zeroCandidates, "\n")
  cat("    forceInner:", forceInner, "\n\n")
print(table(original = a$yInner[change, "original"], rounded = a$yInner[change, "rounded"]))
  cat("--------------------------------------------------\n")
}

## End(Not run)
```

SmallCountData *Function that returns a dataset*

Description

Function that returns a dataset

Usage

```
SmallCountData(dataset, path = NULL)
```

Arguments

dataset	Name of data set within the SmallCountRounding package
path	When non-NULL the data set is read from "path/dataset.RData"

Value

The dataset

Note

Except for "europe6", "eHrc", "eDimList" and "exPSD", the function returns the same datasets as SSBtoolsData.

See Also

SSBtoolsData, Hrc2DimList

Examples

```
SmallCountData("z1")
SmallCountData("e6")
SmallCountData("eHrc")      #  TauArgus coded hierarchies
SmallCountData("eDimList")  #  sdcTable coded hierarchies
SmallCountData("exPSD")   # Example data in presentation at Privacy in statistical databases
```

Index

∗ **print**
print.PLSrounded, 14

aggregate_by_pkg, *7*
AutoHierarchies, *16*

ellipsis::check_dots_used(), *7*
Extend0, *6*, *10*

Formula2ModelMatrix, *7*, *17*
FormulaSelection, *3*, *7*
FormulaSelection.PLSrounded, 3

HD, 3
HDutility, *8*
HDutility *(*HD*)*, 3
Hrc2DimList, *19*

Mipf, *10*
ModelMatrix, *7, 8*, *15*, *17*

PLS2way, 4, *8*
PLSrounding, *2*, *4*, 5, *10–14*, *18*
PLSroundingFits, 10
PLSroundingInner *(*PLSrounding*)*, 5
PLSroundingLoop, 13
PLSroundingPublish *(*PLSrounding*)*, 5
print.PLSrounded, 14

Reduce0exact, *16*
RoundViaDummy, *6–8*, 15
RoundViaDummy(), *6, 7*
RowGroups, *7*

SmallCountData, 19
SmallCountRounding
*(*SmallCountRounding-package*)*, 2
SmallCountRounding-package, 2
SSBtools::ModelMatrix(), *16*
SSBtoolsData, *19*