# Package 'GaussSuppression'

May 22, 2024

**Type** Package

**Title** Tabular Data Suppression using Gaussian Elimination

**Version** 0.8.5

**Date** 2024-05-22

**Maintainer** Øyvind Langsrud <oyl@ssb.no>

**Depends** Matrix

**Imports** SSBtools (>= 1.5.2), RegSDC (>= 0.7.0), stats, methods, utils

**Description** A statistical disclosure control tool to protect tables by suppression using the Gaussian elimination secondary suppression algorithm. A suggestion is to start by working with functions SuppressSmallCounts() and SuppressDominantCells(). These functions use primary suppression functions for the minimum frequency rule and the dominance rule, respectively. Novel functionality for suppression of disclosive cells is also included. General primary suppression functions can be supplied as input to the general working horse function, GaussSuppressionFromData(). Suppressed frequencies can be replaced by synthetic decimal numbers as described in Langsrud (2019) <doi:10.1007/s11222-018-9848-9>.

**License** MIT + file LICENSE

**URL** https://github.com/statisticsnorway/ssb-gausssuppression

**BugReports** https://github.com/statisticsnorway/ssb-gausssuppression/issues

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**Suggests** formattable, kableExtra, knitr, rmarkdown, testthat (>= 3.0.0), lpSolve, Rsymphony, Rglpk, slam, highs

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Øyvind Langsrud [aut, cre],
Daniel Lupp [aut],
Hege Bøvelstad [ctb],

Vidar Norstein Klungre [rev],
Statistics Norway [cph]

**Repository** CRAN

**Date/Publication** 2024-05-22 09:00:03 UTC

# R topics documented:



---

`AdditionalSuppression` *GaussSuppression from data and suppressed data*

---

## Description

Extended version of `GaussSuppressionFromData` that takes into account suppression pattern in suppressed data sent as input

## Usage

```
AdditionalSuppression(
  data,
  ...,
  fun = GaussSuppressionFromData,
  primary = GetDefault(fun, "primary"),
  suppressedData = NULL,
  makePrimary = TRUE,
  makeForced = TRUE,
  forceNotPrimary = TRUE
)
```

## Arguments

| | |
|---|---|
| data | Input data as a data frame |
| ... | Further parameters to GaussSuppressionFromData |
| fun | A function: GaussSuppressionFromData or one of its wrappers such as SuppressSmallCounts and SuppressDominantCells. |
| primary | As input to GaussSuppressionFromData before possible extension caused by suppressedData. Supply NULL if all primary suppressions are retrieved form suppressedData. |
| suppressedData | A data frame or a list of data frames as output from GaussSuppressionFromData. |
| makePrimary | When TRUE, suppression in suppressedData is preserved. |
| makeForced | When TRUE, non-suppression in suppressedData is preserved. An exception is possible primary suppression which has priority over forced. Use forceNotPrimary to avoid this exception. |
| forceNotPrimary | When TRUE, non-suppression in suppressedData is forced to be not primary suppressed. |

## Details

This function is an easy alternative to using PrimaryFromSuppressedData and the relating functions manually. See the examples of PrimaryFromSuppressedData. By default, the suppression pattern in suppressedData is preserved. The behavior can be tuned by the parameters.

Note that the variables used in suppressedData in addition to "suppressed" are those with matching names in crossTable. Others are ignored. See examples (d3, d4, d5). NOW A FIX IS INCLUDED by attribute totCode. EXAMPLES NOT YET CHANGED.

## Value

Aggregated data with suppression information

## Examples

```
z1 <- SSBtoolsData("z1")
z2 <- SSBtoolsData("z2")
z3 <- SSBtoolsData("z3")

# Ordinary suppressions
a <- GaussSuppressionFromData(z1, 1:2, 3, maxN = 5)
b <- GaussSuppressionFromData(z2, 1:4, 5, maxN = 1)

# As b and also suppression pattern in a preserved
b1 <- AdditionalSuppression(z2, 1:4, 5, maxN = 1, suppressedData = a)

# Rows with differences
cbind(b, b1)[b1$suppressed != b$suppressed, ]

# All primary from a
b2 <- AdditionalSuppression(z2, 1:4, 5, suppressedData = a, primary = NULL, singleton = NULL)

# Rows with suppression
b2[b2$suppressed, ]

# All primary from b2
d1 <- AdditionalSuppression(data = z3, 1:6, 7, suppressedData = b2, primary = NULL,
                            singleton = NULL)

# No suppression since no common codes
d1[d1$suppressed, ]

# Use another coding of fylke
z3$fylke_ <- z3$fylke - 4
d2 <- AdditionalSuppression(data = z3, c(1, 3:6, 8), 7, suppressedData = b2, primary = NULL,
                            singleton = NULL)

# Two primary found in b2 -> several secondary
d2[d2$suppressed,]


# Examples demonstrating limitations of AdditionalSuppression
# Variable mnd in suppressedData is not used

# No suppression since unsuppressed rows used by makeForced and forceNotPrimary
d3 <- AdditionalSuppression(data = z3, c(1, 3:4, 8), 7, suppressedData = d2, primary = NULL,
                            singleton = NULL)
d3[d3$suppressed, ]

# Now suppression, but not too much
d4 <- AdditionalSuppression(data = z3, c(1, 3:4, 8), 7, suppressedData = d2,
                            forceNotPrimary = FALSE, primary = NULL, singleton = NULL)
d4[d4$suppressed, ]

# The correct way is to limit the input
```

```
d5 <- AdditionalSuppression(data = z3, c(1, 3:4, 8), 7, suppressedData = d2[d2$mnd == "Total", ],
                            primary = NULL, singleton = NULL)
d5[d5$suppressed, ]
```

---

`CandidatesDefault` *Candidates functions*

---

## Description

Function for GaussSuppressionFromData

## Usage

```
CandidatesDefault(freq, x, secondaryZeros = FALSE, weight, ...)

CandidatesNum(
  secondaryZeros = FALSE,
  freq = NULL,
  num,
  weight,
  x,
  candidatesVar = NULL,
  removeCodes = character(0),
  removeCodesForCandidates = TRUE,
  data,
  charVar,
  ...
)
```

## Arguments

| | |
|---|---|
| `freq` | Vector of output frequencies |
| `x` | The model matrix |
| `secondaryZeros` | When `TRUE`, cells with zero frequency or value are prioritized to be published so that they are not secondary suppressed. This is achieved by this function by having the zero frequency indices first in the retuned order. |
| `weight` | Vector of output weights |
| `...` | Unused parameters |
| `num` | Data frame of output aggregates calculated from `numVar`. When several variables, and without specifying `candidatesVar`, only first is used. |
| `candidatesVar` | One of the variable names from `numVar` to be used in the calculations. Specifying `candidatesVar` helps avoid warnings when multiple `numVar` variables are present. |

| | |
|---|---|
| removeCodes | Same parameter as used in suppression rules, e.g. NContributorsRule. It is often assumed that cells where all contributors (charVar) are present in removeCodes should be published. Here, such cells will be prioritized to achieve this. Note that this functionality is redundant if the same cells are specified by forced. |
| removeCodesForCandidates | removeCodes ignored when set to FALSE. |
| data | Input data as a data frame (needed for removeCodes calculations) |
| charVar | Variable(s) with contributor codes (needed for removeCodes calculations) |

## Details

CandidatesDefault orders the indices decreasingly according to freq or, when weight is non-NULL, (freq+1)*weight. Ties are handled by prioritizing output cells that are calculated from many input cells. In addition, zeros are handled according to parameter secondaryZeros. When freq is negative (special hierarchy), abs(freq)*weight is used.

CandidatesNum orders the indices decreasingly according to absolute values of the numeric variable (according to abs(num[[1]])). In practice this is done by running CandidatesDefault with manipulated weights.

## Value

candidates, GaussSuppression input

---

# ChainedSuppression *Repeated GaussSuppression with forwarding of previous results*

---

## Description

AdditionalSuppression is called several times. Each time with all previous results as suppressedData.

## Usage

```
ChainedSuppression(..., withinArg = NULL)

ChainedSuppressionHi(..., hierarchies)

ChainedSuppressionHi1(..., hierarchies)
```

## Arguments

| | |
|---|---|
| ... | Arguments to AdditionalSuppression/GaussSuppressionFromData that are kept constant. |
| withinArg | A list of named lists. Arguments to AdditionalSuppression/GaussSuppressionFromData that are not kept constant. List elements with suppressed data are also allowed. |

`hierarchies` In the wrapper `ChainedSuppressionHi`, this argument will be used to generate the `withinArg` to `ChainedSuppression` with the same length (see examples). Then, element number `i` of `withinArg` is `list(hierarchies = hierarchies[1:i])`. In the similar wrapper, `ChainedSuppressionHi1`, `withinArg` has always two elements: `list(hierarchies = hierarchies[1])` and `list(hierarchies = hierarchies)`.

## Value

List of data frames. The wrappers, `ChainedSuppressionHi` and `ChainedSuppressionHi1`, return a single data frame, which is the last list item.

## Examples

```
z1 <- SSBtoolsData("z1")
z2 <- SSBtoolsData("z2")
z2b <- z2[3:5]
names(z2b)[1] <- "region"

# As GaussSuppressionFromData when a single element within withinArg
a1 <- ChainedSuppression(z1, 1:2, 3, maxN = 5)
a2 <- ChainedSuppression(z1, withinArg = list(list(dimVar = 1:2, freqVar = 3, maxN = 5)))
identical(a1, a2[[1]])

# b[[3]] include results from b[[1]] and b[[2]]
b <- ChainedSuppression(z1, freqVar = 3, withinArg = list(
       list(dimVar = 1,   maxN = 55),
       list(dimVar = 2,   maxN = 55),
       list(dimVar = 1:2, maxN = 5)))

# d[[2]] is same as b1 in AdditionalSuppression examples
d <- ChainedSuppression(withinArg = list(
       list(data = z1,  dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2,  dimVar = 1:4, freqVar = 5, maxN = 1)))

# Common variable names important.
# Therefore kostragr renamed to region in z2b.
f <- ChainedSuppression(withinArg = list(
       list(data = z1,  dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2b, dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2,  dimVar = 1:4, freqVar = 5, maxN = 1)))

# Parameters so that only suppressions are forwarded.
# This is first iteration in linked tables by iterations.
e <- ChainedSuppression(withinArg = list(
       list(data = z1,  dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2b, dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2,  dimVar = 1:4, freqVar = 5, maxN = 1)),
       makeForced = FALSE, forceNotPrimary = FALSE)

# "A" "annet"/"arbeid" could be suppressed here, but not in f since f[[1]]
```

```
e[[3]][which(e[[3]]$suppressed != f[[3]]$suppressed), ]


#### Demonstrate SuppressionByChainedHierarchies

dimLists <- SSBtools::FindDimLists(z2[, 4:1])


# Two ways of doing the same calculations
g1 <- ChainedSuppressionHi(z2, c(1, 3), 5, maxN = 1, hierarchies = dimLists)
g1b <-  ChainedSuppression(z2, c(1, 3), 5, maxN = 1, withinArg = list(
         list(hierarchies = dimLists[1]),
         list(hierarchies = dimLists[1:2]),
         list(hierarchies = dimLists[1:3])))[[3]]

# Results different after combining hierarchies
g2 <- ChainedSuppressionHi(z2, c(1, 3), 5, maxN = 1,
         hierarchies = SSBtools::AutoHierarchies(dimLists))

# In this case, the same results can be obtained by:
g3 <- ChainedSuppressionHi1(z2, c(1, 3), 5, maxN = 1, hierarchies = dimLists)
```

---

`ComputeIntervals` *Function for calculating intervals for suppressed tables.*

---

## Description

This function solves linear programs to determine interval boundaries for suppressed cells.

## Usage

```
ComputeIntervals(
  x,
  z,
  primary,
  suppressed,
  minVal = NULL,
  lpPackage = "lpSolve",
  gaussI = TRUE,
  allInt = FALSE,
  sparseConstraints = TRUE
)
```

## Arguments

| | |
|---|---|
| x | ModelMatrix, as output from SSBtools::ModelMatrix |
| z | numerical vector with length ncol(x). Corresponds to table cell values |

| | |
|---|---|
| primary | Vector indicating primary suppressed cells. Can be logical or integer. If integer vector, indicates the columns of x which are considered primary suppressed. |
| suppressed | Vector indicating all suppressed cells. Can be logical or integer. If integer vector, indicates the columns of x which are considered suppressed. |
| minVal | a known minimum value for table cells. Default NULL. Note that 'minVal' is interpreted as the limiting value for all suppressed cells. Specifying 'minVal=0' would be redundant, as a minimum value of 0 is anyway assumed for inner cells (see details). |
| lpPackage | The name of the package used to solve linear programs. Currently, 'lpSolve' (default), 'Rsymphony', 'Rglpk' and 'highs' are supported. |
| gaussI | Boolean vector. If TRUE (default), GaussIndependent is used to reduce size of linear program. |
| allInt | Integer variables when TRUE. See `all.int` parameter in `lpSolve` and `types` parameter in `Rsymphony` and `Rglpk`. |
| sparseConstraints | When TRUE, a sparse constraint matrix will be input to the solver. In the case of `lpSolve`, the sparse matrix is represented in triplet form as a dense matrix with three columns, and the `dense.const` parameter is utilized. |

## Details

This function is still experimental.

Default in for `bounds` parameter in `Rsymphony_solve_LP` and `Rglpk_solve_LP`: *The default for each variable is a bound between 0 and* `Inf`. Details in `lpSolve`: *Note that every variable is assumed to be* `>= 0`*!*

## Author(s)

Øyvind Langsrud and Daniel Lupp

---

`FindDominantCells` *Method for finding dominant cells according to (possibly multiple) n,k dominance rules.*

---

## Description

Supports functionality for grouping contributions according to holding variables, as well as calculating dominance in surveys with a given sampling weight. Two methods are implemented, depending on whether the sampling weights sum to total population. The parameter `tauArgusDominance` determines this. If `FALSE`, unweighted contributions are compared to weighted cell values. If `TRUE`, the method described in in the book "Statistical Disclosure Control" (Hundepool et al 2012, p. 151) is used.

## Usage

```
FindDominantCells(
  x,
  inputnum,
  num,
  n,
  k,
  charVar_groups,
  samplingWeight,
  tauArgusDominance = FALSE,
  returnContrib = FALSE
)
```

## Arguments

| | |
|---|---|
| x | model matrix describing relationship between input and published cells |
| inputnum | vector of numeric contributions for each of the input records |
| num | vector of numeric values for each of the published cells |
| n | vector of integers describing n parameters in n,k rules. Must be same length as k parameter. |
| k | vector of numeric values describing k parameters in n,k rules, where percentages are described as numbers less than 100. Must be same length as n parameter. |
| charVar_groups | vector describing which input records should be grouped |
| samplingWeight | vector of sampling weights associated to input records |
| tauArgusDominance | logical value, default FALSE. determines how to handle sampling weights in the dominance rule (see details). |
| returnContrib | logical value, default FALSE. If TRUE return value is the percentage of the first n contributors |

## Value

logical vector describing which publish-cells need to be suppressed.

---

# FixRiskyIntervals *New primary cells to fix risky intervals*

---

## Description

Indices to new primary cells are returned

## Usage

```
FixRiskyIntervals(
  x,
  z,
  primary,
  suppressed,
  candidates = NULL,
  minVal = NULL,
  lpPackage = "lpSolve",
  gaussI = FALSE,
  allInt = FALSE,
  sparseConstraints = TRUE,
  rangeLimits
)
```

## Arguments

| | |
|---|---|
| x | ModelMatrix, as output from SSBtools::ModelMatrix |
| z | numerical vector with length ncol(x). Corresponds to table cell values |
| primary | Vector indicating primary suppressed cells. Can be logical or integer. If integer vector, indicates the columns of x which are considered primary suppressed. |
| suppressed | Vector indicating all suppressed cells. Can be logical or integer. If integer vector, indicates the columns of x which are considered suppressed. |
| candidates | candidates as indices |
| minVal | a known minimum value for table cells. Default NULL. Note that 'minVal' is interpreted as the limiting value for all suppressed cells. Specifying 'minVal=0' would be redundant, as a minimum value of 0 is anyway assumed for inner cells (see details). |
| lpPackage | The name of the package used to solve linear programs. Currently, 'lpSolve' (default), 'Rsymphony', 'Rglpk' and 'highs' are supported. |
| gaussI | Boolean vector. If TRUE (default), GaussIndependent is used to reduce size of linear program. |
| allInt | Integer variables when TRUE. See all.int parameter in lpSolve and types parameter in Rsymphony and Rglpk. |
| sparseConstraints | When TRUE, a sparse constraint matrix will be input to the solver. In the case of lpSolve, the sparse matrix is represented in triplet form as a dense matrix with three columns, and the dense.const parameter is utilized. |
| rangeLimits | As computed by RangeLimitsDefault |

## Details

Code in this function started from a copy of ComputeIntervals

GaussSuppressDec *Cell suppression with synthetic decimal numbers*

## Description

GaussSuppressionFromData is run and decimal numbers are added to output by a modified (for sparse matrix efficiency) version of SuppressDec.

## Usage

```
GaussSuppressDec(
  data,
  ...,
  output = NULL,
  digits = 9,
  nRep = NULL,
  rmse = pi/3,
  sparseLimit = 500,
  rndSeed = 123,
  runIpf = FALSE,
  eps = 0.01,
  iter = 100,
  mismatchWarning = TRUE,
  whenDuplicatedInner = NULL,
  whenMixedDuplicatedInner = warning
)
```

## Arguments

| | |
|---|---|
| data | Input daata as a data frame |
| ... | Further parameters to GaussSuppressionFromData |
| output | NULL (default), "publish", "inner", "publish_inner", or "publish_inner_x" (x also). |
| digits | Parameter to RoundWhole. Values close to whole numbers will be rounded. |
| nRep | NULL or an integer. When >1, several decimal numbers will be generated. |
| rmse | Desired root mean square error of decimal numbers. Variability around the expected, according to the linear model, inner frequencies. The expected frequencies are calculated from the non-suppressed publishable frequencies. |
| sparseLimit | Limit for the number of rows of a reduced x-matrix within the algorithm. When exceeded, a new sparse algorithm is used. |
| rndSeed | If non-NULL, a random generator seed to be used locally within the function without affecting the random value stream in R. |
| runIpf | When TRUE, additional frequencies are generated by iterative proportional fitting using Mipf. |

eps Parameter to Mipf.

iter Parameter to Mipf.

mismatchWarning
Whether to produce the warning "Mismatch between whole numbers and suppression", when relevant. When nRep>1, all replicates must satisfy the whole number requirement for non-suppressed cells. When mismatchWarning is integer (>0), this will be used as parameter digits to RoundWhole when doing mismatch checking (can be quite low when nRep>1).

whenDuplicatedInner
Function to be called when default output and when cells marked as inner correspond to several input cells (aggregated) since they correspond to published cells.

whenMixedDuplicatedInner
Function to be called in the case above when some inner cells correspond to published cells (aggregated) and some not (not aggregated).

## Value

A data frame where inner cells and cells to be published are combined or output according to parameter output.

## Author(s)

Øyvind Langrsud

## Examples

```
z1 <- SSBtoolsData("z1")
GaussSuppressDec(z1, 1:2, 3)
GaussSuppressDec(z1, freqVar = "ant", formula = ~ region + hovedint, maxN = 10)
```

---

# GaussSuppressionFromData

*Cell suppression from input data containing inner cells*

---

## Description

Aggregates are generated followed by primary suppression followed by secondary suppression by Gaussian elimination by GaussSuppression

## Usage

```
GaussSuppressionFromData(
  data,
  dimVar = NULL,
  freqVar = NULL,
  ...,
```

```
  numVar = NULL,
  weightVar = NULL,
  charVar = NULL,
  hierarchies = NULL,
  formula = NULL,
  maxN = suppressWarnings(formals(c(primary)[[1]])$maxN),
  protectZeros = suppressWarnings(formals(c(primary)[[1]])$protectZeros),
  secondaryZeros = suppressWarnings(formals(candidates)$secondaryZeros),
  candidates = CandidatesDefault,
  primary = PrimaryDefault,
  forced = NULL,
  hidden = NULL,
  singleton = SingletonDefault,
  singletonMethod = ifelse(secondaryZeros, "anySumNOTprimary", "anySum"),
  printInc = TRUE,
  output = "publish",
  x = NULL,
  crossTable = NULL,
  preAggregate = is.null(freqVar),
  extraAggregate = preAggregate & !is.null(charVar),
  structuralEmpty = FALSE,
  extend0 = FALSE,
  spec = NULL,
  specLock = FALSE,
  freqVarNew = rev(make.unique(c(names(data), "freq")))[1],
  nUniqueVar = rev(make.unique(c(names(data), "nUnique")))[1],
  forcedInOutput = "ifNonNULL",
  unsafeInOutput = "ifForcedInOutput",
  lpPackage = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | Input data as a data frame |
| `dimVar` | The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified. |
| `freqVar` | A single variable holding counts (name or number). |
| `...` | Further arguments to be passed to the supplied functions and to `ModelMatrix` (such as `inputInOutput` and `removeEmpty`). |
| `numVar` | Other numerical variables to be aggregated |
| `weightVar` | weightVar Weights (costs) to be used to order candidates for secondary suppression |
| `charVar` | Other variables possibly to be used within the supplied functions |
| `hierarchies` | List of hierarchies, which can be converted by `AutoHierarchies`. Thus, the variables can also be coded by `"rowFactor"` or `""`, which correspond to using the categories in the data. |

| | |
|---|---|
| formula | A model formula |
| maxN | Suppression parameter. Cells with frequency <= maxN are set as primary suppressed. Using the default primary function, maxN is by default set to 3. See details. |
| protectZeros | Suppression parameter. When TRUE, cells with zero frequency or value are set as primary suppressed. Using the default primary function, protectZeros is by default set to TRUE. See details. |
| secondaryZeros | Suppression parameter. When TRUE, cells with zero frequency or value are prioritized to be published so that they are not secondary suppressed. Using the default candidates function, secondaryZeros is by default set to FALSE. See details. |
| candidates | GaussSuppression input or a function generating it (see details) Default: CandidatesDefault |
| primary | GaussSuppression input or a function generating it (see details) Default: PrimaryDefault |
| forced | GaussSuppression input or a function generating it (see details) |
| hidden | GaussSuppression input or a function generating it (see details) |
| singleton | GaussSuppression input or a function generating it (see details) Default: SingletonDefault |
| singletonMethod | GaussSuppression input. The default value depends on parameter secondaryZeros which depends on candidates (see details). |
| printInc | GaussSuppression input |
| output | One of "publish" (default), "inner", "publish_inner", "publish_inner_x", "publish_x", "inner_x", "input2functions" (input to supplied functions), "inputGaussSuppression", "inputGaussSuppression_x", "outputGaussSuppression" "outputGaussSuppression_x", "primary", "secondary" and "all". Here "inner" means input data (possibly pre-aggregated) and "x" means dummy matrix (as input parameter x). All input to and output from GaussSuppression, except ..., are returned when "outputGaussSuppression_x". Excluding x and only input are also possible. The code "all" means all relevant output after all the calculations. Currently, this means the same as "publish_inner_x" extended with the matrices (or NULL) xExtraPrimary and unsafe. The former matrix is usually made by KDisclosurePrimary. This latter matrix contains the columns representing unsafe primary suppressions. In addition to x columns corresponding to unsafe in ordinary output (see parameter unsafeInOutput below), possible columns from xExtraPrimary may also be included in the unsafe matrix (see details). |
| x | x (modelMatrix) and crossTable can be supplied as input instead of generating it from ModelMatrix |
| crossTable | See above. |
| 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 and in addition charVar. |
| extraAggregate | When TRUE, the data will be aggregated by the dimensional variables according to dimVar, hierarchies or formula. The aggregated data and the corresponding x-matrix will only be used as input to the singleton function and |

GaussSuppression. This extra aggregation is useful when parameter charVar is used. Supply "publish_inner", "publish_inner_x", "publish_x" or "inner_x" as output to obtain extra aggregated results. Supply "inner" or "input2functions" to obtain other results.

structuralEmpty

When TRUE, output cells with no contributing inner cells (only zeros in column of x) are forced to be not primary suppressed. Thus, these cells are considered as structural zeros. When structuralEmpty is TRUE, the following error message is avoided: Suppressed cells with empty input will not be protected. Extend input data with zeros?. When removeEmpty is TRUE (see "..." below), structuralEmpty is superfluous

extend0 Data is automatically extended by Extend0 when TRUE. Can also be set to "all" which means that input codes in hierarchies are considered in addition to those in data. Parameter extend0 can also be specified as a list meaning parameter varGroups to Extend0.

spec NULL or a named list of arguments that will act as default values.

specLock When TRUE, arguments in spec cannot be changed.

freqVarNew Name of new frequency variable generated when input freqVar is NULL and preAggregate is TRUE. Default is "freq" provided this is not found in names(data).

nUniqueVar Name of variable holding the number of unique contributors. This variable will be generated in the extraAggregate step. Default is "nUnique" provided this is not found in names(data). If an existing variable is passed as input, this variable will apply only when preAggregate/extraAggregate is not done.

forcedInOutput Whether to include forced as an output column. One of "ifNonNULL" (default), "always", "ifany" and "no". In addition, TRUE and FALSE are allowed as alternatives to "always" and "no".

unsafeInOutput Whether to include usafe as an output column. One of "ifForcedInOutput" (default), "always", "ifany" and "no". In addition, TRUE and FALSE are allowed as alternatives to "always" and "no". see details.

lpPackage

- lpPackage**:** When non-NULL, intervals by ComputeIntervals will be included in the output. See its documentation for valid parameter values for 'lpPackage'. If, additionally, at least one of the two RangeLimitsDefault parameters below is specified, further suppression will be performed to satisfy the interval width requirements. Then, the values in the output variable suppressed_integer means: no suppression (0), primary suppression (1), secondary suppression (2), additional suppression applied by an interval algorithm limited to linearly independent cells (3), and further suppression according to the final gauss algorithm (4). Intervals, [lo_1, up_1], are intervals calculated prior to additional suppression.
  - rangePercent**:** Required interval width expressed as a percentage
  - rangeMin**:** Minimum required width of the interval

Please note that interval calculations may have a different interface in future versions.

## Details

The supplied functions for generating GaussSuppression input takes the following arguments: crossTable, x, freq, num, weight, maxN, protectZeros, secondaryZeros, data, freqVar, numVar, weightVar, charVar, dimVar and .... where the two first are ModelMatrix outputs (modelMatrix renamed to x). The vector, freq, is aggregated counts (t(x) %*% data[[freqVar]]). In addition, the supplied singleton function also takes nUniqueVar and (output from) primary as input.

Similarly, num, is a data frame of aggregated numerical variables. It is possible to supply several primary functions joined by c, e.g. (c(FunPrim1, FunPrim2)). All NAs returned from any of the functions force the corresponding cells not to be primary suppressed.

The effect of maxN, protectZeros and secondaryZeros depends on the supplied functions where these parameters are used. Their default values are inherited from the default values of the first primary function (several possible) or, in the case of secondaryZeros, the candidates function. When defaults cannot be inherited, they are set to NULL. In practice the function formals are still used to generate the defaults when primary and/or candidates are not functions. Then NULL is correctly returned, but suppressWarnings are needed.

Singleton handling can be turned off by singleton = NULL or singletonMethod = "none". Both of these choices are identical in the sense that singletonMethod is set to "none" whenever singleton is NULL and vice versa.

Information about uncertain primary suppressions due to forced cells can be found as described by parameters unsafeInOutput and output (= "all"). When forced cells affect singleton problems, this is not implemented. Some information can be seen from warnings. This can also be seen by choosing output = "secondary" together with unsafeInOutput = "ifany" or unsafeInOutput = "always". Then, negative indices from GaussSuppression using unsafeAsNegative = TRUE will be included in the output. Singleton problems may, however, be present even if it cannot be seen as warning/output. In some cases, the problems can be detected by GaussSuppressDec.

In some cases, cells that are forced, hidden, or primary suppressed can overlap. For these situations, forced has precedence over hidden and primary. That is, if a cell is both forced and hidden, it will be treated as a forced cell and thus published. Similarly, any primary suppression of a forced cell will be ignored (see parameter whenPrimaryForced to GaussSuppression). It is, however, meaningful to combine primary and hidden. Such cells will be protected while also being assigned the NA value in the suppressed output variable.

## Value

Aggregated data with suppression information

## Author(s)

Øyvind Langsrud and Daniel Lupp

## Examples

```
z1 <- SSBtoolsData("z1")
GaussSuppressionFromData(z1, 1:2, 3)

z2 <- SSBtoolsData("z2")
```

```
GaussSuppressionFromData(z2, 1:4, 5, protectZeros = FALSE)


# Data as in GaussSuppression examples
df <- data.frame(values = c(1, 1, 1, 5, 5, 9, 9, 9, 9, 9, 0, 0, 0, 7, 7),
                 var1 = rep(1:3, each = 5), var2 = c("A", "B", "C", "D", "E"))

GaussSuppressionFromData(df, c("var1", "var2"), "values")
GaussSuppressionFromData(df, c("var1", "var2"), "values", formula = ~var1 + var2, maxN = 10)
GaussSuppressionFromData(df, c("var1", "var2"), "values", formula = ~var1 + var2, maxN = 10,
     protectZeros = TRUE, # Parameter needed by SingletonDefault and default not in primary
      primary = function(freq, crossTable, maxN, ...)
                  which(freq <= maxN & crossTable[[2]] != "A" & crossTable[, 2] != "C"))

# Combining several primary functions
# Note that NA & c(TRUE, FALSE) equals c(NA, FALSE)
GaussSuppressionFromData(df, c("var1", "var2"), "values", formula = ~var1 + var2, maxN = 10,
       primary = c(function(freq, maxN, protectZeros = TRUE, ...) freq >= 45,
                   function(freq, maxN, ...) freq <= maxN,
                   function(crossTable, ...) NA & crossTable[[2]] == "C",
                   function(crossTable, ...) NA & crossTable[[1]]== "Total"
                                              & crossTable[[2]]== "Total"))

# Similar to GaussSuppression examples
GaussSuppressionFromData(df, c("var1", "var2"), "values", formula = ~var1 * var2,
       candidates = NULL, singleton = NULL, protectZeros = FALSE, secondaryZeros = TRUE)
GaussSuppressionFromData(df, c("var1", "var2"), "values", formula = ~var1 * var2,
       singleton = NULL, protectZeros = FALSE, secondaryZeros = FALSE)
GaussSuppressionFromData(df, c("var1", "var2"), "values", formula = ~var1 * var2,
       protectZeros = FALSE, secondaryZeros = FALSE)


# Examples with zeros as singletons
z <- data.frame(row = rep(1:3, each = 3), col = 1:3, freq = c(0, 2, 5, 0, 0, 6:9))
GaussSuppressionFromData(z, 1:2, 3, singleton = NULL)
GaussSuppressionFromData(z, 1:2, 3, singletonMethod = "none") # as above
GaussSuppressionFromData(z, 1:2, 3)
GaussSuppressionFromData(z, 1:2, 3, protectZeros = FALSE, secondaryZeros = TRUE, singleton = NULL)
GaussSuppressionFromData(z, 1:2, 3, protectZeros = FALSE, secondaryZeros = TRUE)
```

---

GaussSuppressionTwoWay

*Two-way iteration variant of* GaussSuppressionFromData

---

## Description

Internally, data is organized in a two-way table.

Use parameter `colVar` to choose hierarchies for columns (others will be rows). Iterations start by column by column suppression. The algorithm utilizes `HierarchyCompute2`.

With two-way iterations, larger data can be handled, but there is a residual risk. The method is a special form of linked-table iteration. Separately, the rows and columns are protected by GaussSuppression and they have common suppressed cells.

## Usage

```
GaussSuppressionTwoWay(
  data,
  dimVar = NULL,
  freqVar = NULL,
  numVar = NULL,
  weightVar = NULL,
  charVar = NULL,
  hierarchies,
  formula = NULL,
  maxN = suppressWarnings(formals(c(primary)[[1]])$maxN),
  protectZeros = suppressWarnings(formals(c(primary)[[1]])$protectZeros),
  secondaryZeros = suppressWarnings(formals(candidates)$secondaryZeros),
  candidates = CandidatesDefault,
  primary = PrimaryDefault,
  forced = NULL,
  hidden = NULL,
  singleton = SingletonDefault,
  singletonMethod = ifelse(secondaryZeros, "anySumNOTprimary", "anySum"),
  printInc = TRUE,
  output = "publish",
  preAggregate = is.null(freqVar),
  colVar = names(hierarchies)[1],
  removeEmpty = TRUE,
  inputInOutput = TRUE,
  candidatesFromTotal = TRUE,
  structuralEmpty = FALSE,
  freqVarNew = rev(make.unique(c(names(data), "freq")))[1],
  ...
)
```

## Arguments

| | |
|---|---|
| data | Input data as a data frame |
| dimVar | The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified. |
| freqVar | A single variable holding counts (name or number). |
| numVar | Other numerical variables to be aggregated |
| weightVar | weightVar Weights (costs) to be used to order candidates for secondary suppression |
| charVar | Other variables possibly to be used within the supplied functions |

| | |
|---|---|
| hierarchies | List of hierarchies, which can be converted by AutoHierarchies. Thus, the variables can also be coded by "rowFactor" or "", which correspond to using the categories in the data. |
| formula | A model formula |
| maxN | Suppression parameter. See GaussSuppressionFromData. |
| protectZeros | Suppression parameter. See GaussSuppressionFromData. |
| secondaryZeros | Suppression parameter. See GaussSuppressionFromData. |
| candidates | GaussSuppression input or a function generating it (see details) Default: CandidatesDefault |
| primary | GaussSuppression input or a function generating it (see details) Default: PrimaryDefault |
| forced | GaussSuppression input or a function generating it (see details) |
| hidden | GaussSuppression input or a function generating it (see details) |
| singleton | NULL or a function generating GaussSuppression input (logical vector not possible) Default: SingletonDefault |
| singletonMethod | GaussSuppression input |
| printInc | GaussSuppression input |
| output | One of "publish" (default), "inner". Here "inner" means input data (possibly pre-aggregated). |
| 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 and in addition charVar. |
| colVar | Hierarchy variables for the column groups (others in row group). |
| removeEmpty | When TRUE (default) empty output corresponding to empty input is removed. When NULL, removal only within the algorithm (x matrices) so that such empty outputs are never secondary suppressed. |
| inputInOutput | Logical vector (possibly recycled) for each element of hierarchies. TRUE means that codes from input are included in output. Values corresponding to "rowFactor" or "" are ignored. |
| candidatesFromTotal | When TRUE (default), same candidates for all rows and for all columns, computed from row/column totals. |
| structuralEmpty | See GaussSuppressionFromData. |
| freqVarNew | Name of new frequency variable generated when input freqVar is NULL and preAggregate is TRUE. Default is "freq" provided this is not found in names(data). |
| ... | Further arguments to be passed to the supplied functions. |

## Details

The supplied functions for generating GaussSuppression input behave as in GaussSuppressionFromData with some exceptions. When candidatesFromTotal is TRUE (default) the candidate function will be run locally once for rows and once for columns. Each time based on column or row totals. The

global x-matrix will only be generated if one of the functions supplied needs it. Non-NULL singleton can only be supplied as a function. This function will be run locally within the algorithm before each call to GaussSuppression.

Note that a difference from GaussSuppressionFromData is that parameter removeEmpty is set to TRUE by default.

Another difference is that duplicated combinations is not allowed. Normally duplicates are avoided by setting preAggregate to TRUE. When the charVar parameter is used, this can still be a problem. See the examples for a possible workaround.

## Value

Aggregated data with suppression information

## Examples

```
z3 <- SSBtoolsData("z3")

dimListsA <- SSBtools::FindDimLists(z3[, 1:6])
dimListsB <- SSBtools::FindDimLists(z3[, c(1, 4, 5)])

set.seed(123)
z <- z3[sample(nrow(z3),250),]

## Not run:
out1 <- GaussSuppressionTwoWay(z, freqVar = "ant", hierarchies = dimListsA,
                               colVar = c("hovedint"))

## End(Not run)
out2 <- GaussSuppressionTwoWay(z, freqVar = "ant", hierarchies = dimListsA,
                               colVar = c("hovedint", "mnd"))
out3 <- GaussSuppressionTwoWay(z, freqVar = "ant", hierarchies = dimListsB,
                               colVar = c("region"))
out4 <- GaussSuppressionTwoWay(z, freqVar = "ant", hierarchies = dimListsB,
                               colVar = c("hovedint", "region"))

# "mnd" not in  hierarchies -> duplicated combinations in input
# Error when  preAggregate is FALSE: Index method failed. Duplicated combinations?
out5 <- GaussSuppressionTwoWay(z, freqVar = "ant", hierarchies = dimListsA[1:3],
                        protectZeros = FALSE, colVar = c("hovedint"), preAggregate = TRUE)


# charVar needed -> Still problem when preAggregate is TRUE
# Possible workaround by extra hierarchy
out6 <- GaussSuppressionTwoWay(z, freqVar = "ant", charVar = "mnd2",
                         hierarchies = c(dimListsA[1:3], mnd2 = "Total"), # include charVar
                              inputInOutput = c(TRUE, TRUE, FALSE),  # FALSE -> only Total
                               protectZeros = FALSE, colVar = c("hovedint"),
                               preAggregate = TRUE,
                               hidden = function(x, data, charVar, ...)
                        as.vector((t(x) %*% as.numeric(data[[charVar]] == "M06M12")) == 0))
```

`KDisclosurePrimary` *Construct primary suppressed difference matrix*

## Description

Function for constructing model matrix columns representing primary suppressed difference cells

## Usage

```
KDisclosurePrimary(
  data,
  x,
  crossTable,
  freqVar,
  mc_hierarchies = NULL,
  coalition = 1,
  upper_bound = Inf,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | a data.frame representing the data set |
| `x` | ModelMatrix generated by parent function |
| `crossTable` | crossTable generated by parent function |
| `freqVar` | name of the frequency variable in `data` |
| `mc_hierarchies` | a hierarchy representing meaningful combinations to be protected. Default value is `NULL`. |
| `coalition` | numeric vector of length one, representing possible size of an attacking coalition. This parameter corresponds to the parameter k in the definition of k-disclosure. |
| `upper_bound` | numeric value representing minimum count considered safe. Default set to `Inf` |
| `...` | parameters passed to children functions |

## Value

dgCMatrix corresponding to primary suppressed cells

## Author(s)

Daniel P. Lupp

LazyLinkedTables *Linked tables by full* GaussSuppressionFromData *iterations*

## Description

AdditionalSuppression is called several times as in ChainedSuppression

## Usage

```
LazyLinkedTables(..., withinArg = NULL, maxIterLinked = 1000)
```

## Arguments

| | |
|---|---|
| ... | Arguments to GaussSuppressionFromData that are kept constant. |
| withinArg | A list of named lists. Arguments to GaussSuppressionFromData that are not kept constant. |
| maxIterLinked | Maximum number of GaussSuppressionFromData calls for each table. |

## Details

This function is created as a spin-off from AdditionalSuppression and ChainedSuppression. The calculations run GaussSuppressionFromData from the input each time. There is no doubt that this can be done more efficiently.

A consequence of this lazy implementation is that, in output, primary and suppressed are identical.

Note that there is a residual risk when suppression linked tables by iterations.

## Value

List of data frames

## Note

In this function, the parameters makeForced and forceNotPrimary to AdditionalSuppression are forced to be FALSE.

## Examples

```
z1 <- SSBtoolsData("z1")
z2 <- SSBtoolsData("z2")
z2b <- z2[3:5]  # As in ChainedSuppression example
names(z2b)[1] <- "region"

# The two region hierarchies as two linked tables
a <- LazyLinkedTables(z2, freqVar = 5, withinArg = list(
       list(dimVar = c(1, 2, 4)),
```

```
       list(dimVar = c(1, 3, 4))))

# As 'f' and 'e' in ChainedSuppression example.
# 'A' 'annet'/'arbeid' suppressed in b[[1]], since suppressed in b[[3]].
b <- LazyLinkedTables(withinArg = list(
       list(data = z1, dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2b, dimVar = 1:2, freqVar = 3, maxN = 5),
       list(data = z2, dimVar = 1:4, freqVar = 5, maxN = 1)))
```

---

MagnitudeRule *Dominance* (n,k) *or p% rule for magnitude tables*

---

## Description

Supports application of multiple values for n and k. The function works on magnitude tables containing negative cell values by calculating contribution based on absolute values.

## Usage

```
MagnitudeRule(
  data,
  x,
  numVar,
  n = NULL,
  k = NULL,
  pPercent = NULL,
  protectZeros = FALSE,
  charVar = NULL,
  removeCodes = character(0),
  sWeightVar = NULL,
  domWeightMethod = "default",
  allDominance = FALSE,
  outputWeightedNum = !is.null(sWeightVar),
  dominanceVar = NULL,
  ...
)

DominanceRule(data, n, k, protectZeros = FALSE, ...)

PPercentRule(data, pPercent, protectZeros = FALSE, ...)
```

## Arguments

| | |
|---|---|
| data | the dataset |
| x | ModelMatrix generated by parent function |
| numVar | vector containing numeric values in the data set |

| | |
|---|---|
| n | Parameter n in dominance rule. |
| k | Parameter k in dominance rule. |
| pPercent | Parameter in the p% rule, when non-NULL. Parameters n and k will then be ignored. Technically, calculations are performed internally as if n = 1:2. The results of these intermediate calculations can be viewed by setting allDominance = TRUE. |
| protectZeros | parameter determining whether cells with value 0 should be suppressed. |
| charVar | Variable in data holding grouping information. Dominance will be calculated after aggregation within these groups. |
| removeCodes | A vector of charVar codes that are to be excluded when calculating dominance percentages. Essentially, the corresponding numeric values from dominanceVar or numVar are set to zero before proceeding with the dominance calculations. With empty charVar row indices are assumed and conversion to integer is performed. |
| sWeightVar | variable with sampling weights to be used in dominance rule |
| domWeightMethod | character representing how weights should be treated in the dominance rule. See Details. |
| allDominance | Logical parameter. If TRUE, adds primary columns for each pair of parameters n,k in the dominance rules |
| outputWeightedNum | logical value to determine whether weighted numerical value should be included in output. Default is TRUE if sWeightVar is provided. |
| dominanceVar | When specified, dominanceVar is used in place of numVar. Specifying dominanceVar is beneficial for avoiding warnings when there are multiple numVar variables. Typically, dominanceVar will be one of the variables already included in numVar. |
| ... | unused parameters |

## Details

This method only supports suppressing a single numeric variable. There are multiple ways of handling sampling weights in the dominance rule. the default method implemented here compares unweighted sample values with the corresponding weighted cell totals. if domWeightMethod is set to "tauargus", the method implemented in tauArgus is used. For more information on this method, see "Statistical Disclosure Control" by Hundepool et al (2012, p. 151).

## Value

logical vector that is TRUE in positions corresponding to cells breaching the dominance rules.

## Note

Explicit protectZeros in wrappers since default needed by GaussSuppressionFromData

## Author(s)

Daniel Lupp and Øyvind Langsrud

## Examples

```
  set.seed(123)
z <- SSBtools::MakeMicro(SSBtoolsData("z2"), "ant")
z$value <- sample(1:1000, nrow(z), replace = TRUE)

GaussSuppressionFromData(z, dimVar = c("region", "fylke", "kostragr", "hovedint"),
numVar = "value", candidates = CandidatesNum, primary = DominanceRule, preAggregate = FALSE,
singletonMethod = "sub2Sum", n = c(1, 2), k = c(65, 85), allDominance = TRUE)


num <- c(100,
         90, 10,
         80, 20,
         70, 30,
         50, 25, 25,
         40, 20, 20, 20,
         25, 25, 25, 25)
v1 <- c("v1",
        rep(c("v2", "v3", "v4"), each = 2),
        rep("v5", 3),
        rep(c("v6", "v7"), each = 4))
sw <- c(1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1)
d <- data.frame(v1 = v1, num = num, sw = sw)

# without weights
GaussSuppressionFromData(d, formula = ~v1 - 1,
 numVar = "num",  n = c(1,2), k = c(80,70),
  preAggregate = FALSE, allDominance = TRUE, candidates = CandidatesNum,
  primary = DominanceRule)

# with weights, standard method
GaussSuppressionFromData(d, formula = ~v1 - 1,
 numVar = "num",  n = c(1,2), k = c(80,70), sWeightVar = "sw",
 preAggregate = FALSE, allDominance = TRUE, candidates = CandidatesNum,
 primary = DominanceRule)

# with weights, tauargus method
GaussSuppressionFromData(d, formula = ~v1 - 1,
 numVar = "num",  n = c(1,2), k = c(80,70), sWeightVar = "sw",
 preAggregate = FALSE, allDominance = TRUE, candidates = CandidatesNum,
 primary = DominanceRule, domWeightMethod = "tauargus")
```

---

`MaxContribution` *Find major contributions to aggregates*

---

## Description

Assuming aggregates are calculated via a dummy matrix by `z = t(x) %*% y`, the `n` largest contributions are found (value or index) for each aggregate.

## Usage

```
MaxContribution(
  x,
  y,
  n = 1,
  decreasing = TRUE,
  index = FALSE,
  groups = NULL,
  return2 = FALSE
)
```

## Arguments

| | |
|---|---|
| x | A (sparse) dummy matrix |
| y | Vector of input values (contributors) |
| n | Number of contributors to be found |
| decreasing | Ordering parameter. Smallest contributors found when FALSE. |
| index | Indices to y returned when TRUE |
| groups | When non-NULL, major contributions after aggregation within groups. Cannot be combined with index = TRUE. The missing group category is excluded. |
| return2 | When TRUE, two matrices are returned, value and id. The latter contains indices when group is NULL and otherwise a character matrix of groups. |

## Value

Matrix with lagest contributions in first column, second largest in second column and so on. Alternative output when using parameters index or return2.

## Author(s)

Øyvind Langsrud

## See Also

ModelMatrix

## Examples

```
library(SSBtools)

z <- SSBtoolsData("sprt_emp_withEU")
z$age[z$age == "Y15-29"] <- "young"
z$age[z$age == "Y30-64"] <- "old"

a <- ModelMatrix(z, formula = ~age + geo, crossTable = TRUE)

cbind(as.data.frame(a$crossTable), MaxContribution(a$modelMatrix, z$ths_per, 1))
cbind(a$crossTable, MaxContribution(a$modelMatrix, z$ths_per, 10))
```

```
cbind(a$crossTable, MaxContribution(a$modelMatrix, z$ths_per, 10, index = TRUE))

# Both types of output can be achieved with return2 = TRUE)
identical(MaxContribution(a$modelMatrix, z$ths_per, 10, return2 = TRUE),
          list(value =  MaxContribution(a$modelMatrix, z$ths_per, 10),
               id =  MaxContribution(a$modelMatrix, z$ths_per, 10, index = TRUE)))

b <- ModelMatrix(z[, -4], crossTable = TRUE, inputInOutput = c(TRUE, FALSE, TRUE))

k <- cbind(b$crossTable, MaxContribution(b$modelMatrix, z$ths_per, 10))

gr18 <- paste0("g", 1:18)                          # Each row is a group
k18 <- cbind(b$crossTable, MaxContribution(b$modelMatrix, z$ths_per, 10, groups = gr18))
identical(k, k18) # TRUE

gr9 <- paste0("g", as.integer(10 * z$ths_per)%%10) # 9 groups from decimal
k9 <- cbind(b$crossTable, MaxContribution(b$modelMatrix, z$ths_per, 10, groups = gr9))

k18[c(4, 13, 17, 33), ]
k9[c(4, 13, 17, 33), ]

# Group info obtained with return2 = TRUE
k9_id <- cbind(b$crossTable, MaxContribution(b$modelMatrix, z$ths_per, 10, groups = gr9,
                                             return2 = TRUE)$id)
k9_id[c(4, 13, 17, 33), ]


# Verify similarity
z$y <- z$ths_per + (1:nrow(z))/100  # to avoid equal values
id1 <- MaxContribution(b$modelMatrix, z$y, 10, index = TRUE)
id1[!is.na(id1)] <- paste0("g", id1[!is.na(id1)])
mc2 <- MaxContribution(b$modelMatrix, z$y, 10, groups = gr18, return2 = TRUE)
id2 <- mc2$id
identical(id1, id2)
```

---

`Ncontributors` *Find the number of unique groups contributing to aggregates*

---

## Description

Assuming aggregates are calculated via a dummy matrix by `z = t(x) %*% y`, the the number of unique contributing groups, according to a grouping variable, are found for each aggregate. The missing group category is not counted.

## Usage

```
Ncontributors(x, groups)
```

### Arguments

| | |
|---|---|
| x | A (sparse) dummy matrix |
| groups | Vector of group categories |

### Value

Vector of numbers of unique groups

### Author(s)

Øyvind Langsrud

### See Also

ModelMatrix

### Examples

```
library(SSBtools)

z <- SSBtoolsData("sprt_emp_withEU")
z$age[z$age == "Y15-29"] <- "young"
z$age[z$age == "Y30-64"] <- "old"
z$groups <- c("A", "A", "B", "A", "B", "C")

a <- ModelMatrix(z, formula = ~age*eu + geo + year, crossTable = TRUE)

cbind(as.data.frame(a$crossTable), nGroups = Ncontributors(a$modelMatrix, z$groups))
cbind(as.data.frame(a$crossTable), nYears = Ncontributors(a$modelMatrix, z$year))
cbind(as.data.frame(a$crossTable), nUnique_ths_per = Ncontributors(a$modelMatrix, z$ths_per))
```

---

## NcontributorsHolding *Ncontributors with holding-indicator*

---

### Description

The aggregates (columns of x) are grouped by a holding indicator. Within each holding group, the number of unique groups (output) is set to be equal.

### Usage

```
NcontributorsHolding(x, groups, holdingInd = NULL)
```

### Arguments

| | |
|---|---|
| x | A (sparse) dummy matrix |
| groups | Vector of group categories |
| holdingInd | Vector of holding group categories |

## Details

A representative within the holding group is used to calculate output by Ncontributors. The one with maximal column sum of x is chosen as the representative. Normally this will be an aggregate representing the holding group total. When holdingInd is NULL (default), the function is equivalent to Ncontributors.

## Value

Vector of numbers of unique groups

## Author(s)

Øyvind Langsrud

---

# NContributorsRule *Number of contributors suppression rule*

---

## Description

The number of contributors is the number unique contributing 'charVar' codes.

## Usage

```
NContributorsRule(
  data,
  freq,
  numVar,
  x,
  maxN = 3,
  protectZeros = FALSE,
  charVar = NULL,
  removeCodes = character(0),
  remove0 = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| data | Input data as a data frame |
| freq | Vector of aggregate frequencies |
| numVar | Numerical variables. When several variables, only first is used. |
| x | Model matrix generated by parent function |
| maxN | Primary suppression when number of contributors <= maxN. |
| protectZeros | Suppression parameter. Only TRUE (default) is used implemented. |

| | |
|---|---|
| charVar | Variable(s) with contributor codes. When empty, unique contributor in each row is assumed. When several variables, see details. |
| removeCodes | Vector of codes to be omitted when counting contributors. With empty charVar row indices are assumed and conversion to integer is performed. |
| remove0 | When set to TRUE (default), data rows in which the first numVar (if any) is zero are excluded from the count of contributors. Alternatively, remove0 can be specified as one or more variable names. In this case, all data rows with a zero in any of the specified variables are omitted from the contributor count. Specifying remove0 as variable name(s) is useful for avoiding warning when there are multiple numVar variables. |
| ... | unused parameters |

## Details

When several charVar variables, the rule is applied independently to each variable. Primary suppression in at least one case results in primary suppression in the output. It is possible to specify maxN and removeCodes independently for each charVar by using a named list as input with charVar as names. E.g. maxN = list(char1 = 3, char2 = 2).

## Value

List where first element is logical vector defining primary suppressions. The second element is data frame where nRule is number contributors used in rule and where nAll is similar, but without omitting codes in removeCodes.

---

# PackageSpecs *Function for viewing built-in GaussSuppression specs*

---

## Description

Functions to retrieve the built-in specs. These can be retrieved using either numerical indices or by specifying the spec name, see Details.

## Usage

```
PackageSpecs(x = NULL, printTable = FALSE)
```

## Arguments

| | |
|---|---|
| x | the character name or index of the spec to be returned. If NULL (default), returns list of all specs |
| printTable | Logical value (default FALSE). If TRUE, prints a table description of all specs. Primarily used for documentation purposes. |

## Details

The following table summarizes the built-in specs. Columns represent different specs, and rows represent the parameter settings.

| | smallCountSpec | dominanceSpec | fewContributorsSpec | kDisclosureSpec |
|---|---|---|---|---|
| **primary** | PrimaryDefault | MagnitudeRule | NContributorsRule | KDisclosurePrima |
| **protectZeros** | TRUE | FALSE | FALSE | FALSE |
| **candidates** | CandidatesDefault | CandidatesNum | CandidatesNum | DirectDisclosureC |
| **singleton** | SingletonDefault | SingletonUniqueContributor | SingletonUniqueContributor | SingletonDefault |
| **extend0** | TRUE | FALSE | FALSE | TRUE |
| **preAggregate** | is.null(freqVar) | !is.null(charVar) | !is.null(charVar) | is.null(freqVar) |
| **extraAggregate** | FALSE | TRUE | TRUE | FALSE |
| **secondaryZeros** | FALSE | FALSE | FALSE | 1 |
| **domWeightMethod** | | "default" | | |
| **singletonMethod** | | "numttHTT" | "numttHTT" | "anySumNOTprin |

## Value

returns a spec (if `!is.null(x)`), list of all specs (if `is.null(x)` and `printTable = FALSE`), or markdown table describing all specs (if `printTable = TRUE`).

## Examples

```
PackageSpecs()
PackageSpecs(1)
PackageSpecs("smallCountSpec")
PackageSpecs(printTable = TRUE)
```

---

`PrimaryDefault` *Default primary function*

---

## Description

Function for `GaussSuppressionFromData`

## Usage

```
PrimaryDefault(freq, maxN = 3, protectZeros = TRUE, ...)
```

## Arguments

| | |
|---|---|
| `freq` | Vector of output frequencies |
| `maxN` | Cells with frequency `<= maxN` are set as primary suppressed. |
| `protectZeros` | When `TRUE`, cells with zero frequency are set as primary suppressed. |
| `...` | Unused parameters |

## Value

primary, `GaussSuppression` input

PrimaryFromSuppressedData
primary *and* forced *from suppressed data*

## Description

Function for GaussSuppressionFromData

## Usage

```
PrimaryFromSuppressedData(
  x,
  crossTable,
  suppressedData,
  forcedData = FALSE,
  totCode = FindTotCode2(x, crossTable),
  ...
)

ForcedFromSuppressedData(..., forcedData = TRUE)

NotPrimaryFromSuppressedData(..., forcedData = TRUE)
```

## Arguments

| | |
|---|---|
| x | A (sparse) dummy matrix |
| crossTable | crossTable generated by parent function |
| suppressedData | A data frame or a list of data frames as output from GaussSuppressionFromData. If the variable suppressed is not included, all rows are considered suppressed. |
| forcedData | When TRUE, the suppressed coding is swapped. |
| totCode | A named list of totals codes |
| ... | Unused parameters |

## Details

ForcedFromSuppressedData uses forcedData = TRUE and hence a vector to be use as forced is generated. NotPrimaryFromSuppressedData is similar, but TRUE elements are replaced by NA's. Hence the result can be used as an extra primary vector to ensure that code combinations not suppressed according to suppressedData are forced not to be primary suppressed.

The variables used in suppressedData in addition to "suppressed" are those with matching names in crossTable. Others are ignored. For variables in crossTable not in suppressedData, only totals are considered. Others rows are ignored when mathing with suppressedData.

When suppressedData is a list, the final result is the union of individual results of each data frame.

## Value

Logical vector to be used as GaussSuppression input

## Examples

```
z2 <- SSBtoolsData("z2")

# Data to be used as suppressedData
a <- GaussSuppressionFromData(z2, c(1, 3, 4), 5, protectZeros = FALSE)

# For alternative ways to suppress the same table
b1 <- GaussSuppressionFromData(z2, 1:4, 5)
b2 <- GaussSuppressionFromData(z2, 1:4, 5, primary = c(PrimaryDefault, PrimaryFromSuppressedData),
                               suppressedData = a)
b3 <- GaussSuppressionFromData(z2, 1:4, 5, primary = c(PrimaryDefault, PrimaryFromSuppressedData),
                               suppressedData = a, forced = ForcedFromSuppressedData)
b4 <- GaussSuppressionFromData(z2, 1:4, 5,
        primary = c(PrimaryDefault, PrimaryFromSuppressedData, NotPrimaryFromSuppressedData),
                               suppressedData = a, forced = ForcedFromSuppressedData)

# Reducing data to rows mathing a
b1r <- b1[SSBtools::Match(a[1:2], b1[1:2]), ]
b2r <- b2[SSBtools::Match(a[1:2], b2[1:2]), ]
b3r <- b3[SSBtools::Match(a[1:2], b3[1:2]), ]
b4r <- b4[SSBtools::Match(a[1:2], b4[1:2]), ]


# Look at rows where new suppression is different from that in a

# Both TRUE and FALSE changed
cbind(a, b1r)[b1r$suppressed != a$suppressed, c(1:5, 9:10)]

# Only  FALSE changed to TRUE (suppression is preserved)
cbind(a, b2r)[b2r$suppressed != a$suppressed, c(1:5, 9:10)]

# Only change is due to new primary suppression rule (protectZeros = TRUE)
cbind(a, b3r)[b3r$suppressed != a$suppressed, c(1:5, 9:10)]

# No changes
cbind(a, b4r)[b4r$suppressed != a$suppressed, c(1:5, 9:10)]
```

---

# PrimaryRemoveWg *Special functions for the avoidance of suppression*

---

## Description

The SSBtools function WildcardGlobbing is utilized

## Usage

```
PrimaryRemoveWg(wg = NULL, ..., crossTable)

CandidatesNumWg(wg = NULL, ..., crossTable)

ForcedWg(crossTable, wg = NULL, ...)
```

## Arguments

| | |
|---|---|
| wg | data.frame with wildcard/globbing. A parameter to WildcardGlobbing |
| ... | unused parameters |
| crossTable | crossTable generated by parent function |

## Details

CandidatesNumWg is a generalization of CandidatesNumWg

## Value

logical vector or row indices

## Examples

```
dataset <- SSBtoolsData("magnitude1")

a1 <- SuppressDominantCells(data = dataset, numVar = "value",
       dimVar = c("sector4", "geo"), n = 1:2, k = c(77, 99))

a1

wg <- data.frame(sector4 = "Ind*", geo = c("Ice????", "Portugal"))
wg

# Industry:Portugal not primary, but suppressed
a2 <- SuppressDominantCells(data = dataset, numVar = "value",
       dimVar = c("sector4", "geo"), n = 1:2, k = c(77, 99),
       wg = wg, primary = c(DominanceRule, PrimaryRemoveWg))

a2

# Industry:Portugal not primary and not suppressed
a3 <- SuppressDominantCells(data = dataset, numVar = "value",
       dimVar = c("sector4", "geo"), n = 1:2, k = c(77, 99),
       wg = wg, primary = c(DominanceRule, PrimaryRemoveWg),
       candidates = CandidatesNumWg)
a3

# Industry:Portugal primary, but not suppressed
a4 <- SuppressDominantCells(data = dataset, numVar = "value",
       dimVar = c("sector4", "geo"), n = 1:2, k = c(77, 99),
       wg = wg, forced = ForcedWg, whenPrimaryForced = message)
```

a4

---

RangeLimitsDefault *Default range limit function*

---

## Description

Preliminary function

## Usage

```
RangeLimitsDefault(
  ...,
  rangePercent = 0,
  rangeMin = 0,
  primary,
  num,
  freq,
  freqVar,
  dominanceVar = NULL,
  intervalVar = NULL
)
```

## Arguments

| | |
|---|---|
| ... | Unused parameters |
| rangePercent | Required interval width expressed as a percentage |
| rangeMin | Minimum required width of the interval |
| primary | primary |
| num | num |
| freq | freq |
| freqVar | freqVar |
| dominanceVar | dominanceVar |
| intervalVar | Numerical variable(s) for interval calculations. When NULL, dominanceVar, first numVar or freqVar will be used. |

## Value

matrix with named columns

## Examples

```
dat <- SSBtoolsData("magnitude1")
dat["num2"] <- 1:nrow(dat)

SuppressDominantCells(data = dat,
    numVar = "value",
    formula = ~sector2 * geo + sector4 * eu,
    contributorVar = "company",
    n = 1:2, k = c(80, 99),
    output = RangeOutputFunction, rangePercent = 10, rangeMin = 1)

SuppressDominantCells(data = dat,
    numVar = c("value", "num2"),
    formula = ~sector2 * geo + sector4 * eu,
    contributorVar = "company",
    n = 1:2, k = c(80, 99),
    output = RangeOutputFunction,
    intervalVar = c("value","freq", "num2"),
    rangePercent = c(10, 10, 30), rangeMin = c(1, 0.2222, 2.222))
```

---

# SingletonDefault *Default singleton function*

---

## Description

Function for GaussSuppressionFromData

## Usage

```
SingletonDefault(data, freqVar, protectZeros, secondaryZeros, ...)
```

## Arguments

| | |
|---|---|
| data | Input data, possibly pre-aggregated within GaussSuppressionFromData |
| freqVar | A single variable holding counts (input to GaussSuppressionFromData) |
| protectZeros | Suppression parameter (see GaussSuppressionFromData) |
| secondaryZeros | Suppression parameter (see GaussSuppressionFromData) |
| ... | Unused parameters |

## Details

This function marks input cells as singletons according to the input frequencies (freqVar). Zero frequencies are set to singletons when protectZeros or secondaryZeros is TRUE. Otherwise, ones are set to singletons. Empty freqVar is treated as all frequencies being ones.

## Value

singleton, GaussSuppression input

SingletonUniqueContributor

*Unique contributor singleton function*

## Description

Function for GaussSuppressionFromData

## Usage

```
SingletonUniqueContributor(
  data,
  freqVar = NULL,
  nUniqueVar = NULL,
  charVar = NULL,
  removeCodes = character(0),
  integerSingleton = length(charVar) > 0,
  x,
  primary = integer(0),
  whenPrimaryMatters = warning,
  whenNoVar = TRUE,
  specialMultiple = TRUE,
  ...
)

SingletonUniqueContributor0(data, numVar, dominanceVar = NULL, ...)
```

## Arguments

| | |
|---|---|
| data | Input data, possibly pre-aggregated within GaussSuppressionFromData |
| freqVar | A single variable holding counts (input to GaussSuppressionFromData) |
| nUniqueVar | A single variable holding the number of unique contributors. |
| charVar | Variable with contributor codes. |
| removeCodes | Vector, list or data frame of codes considered non-singletons. Single element lists and single column data frames behave just like vectors. In other cases, charVar-names must be used. With empty charVar a vector of row indices is assumed and conversion to integer is performed. See examples. |
| integerSingleton | Integer output when TRUE. See details. |
| x | ModelMatrix generated by parent function |
| primary | Vector (integer or logical) specifying primary suppressed cells. It will be ensured that any non-suppressed inner cell is not considered a singleton. |
| whenPrimaryMatters | Function to be called when primary caused non-singleton. Supply NULL to do nothing. |

| | |
|---|---|
| whenNoVar | When TRUE, and without nUniqueVar and freqVar in input, all cells will be marked as singletons. |
| specialMultiple | When TRUE, and when integerSingleton & length(charVar) > 1 & length(nUniqueVar), a special method is used. By re-coding to single charVar and by re-calculating nUnique. To be unique (nUnique=1), uniqueness is only required for a single charvar. Otherwise, the charvar combination must be unique. |
| ... | Unused parameters |
| numVar | vector containing numeric values in the data set |
| dominanceVar | When specified, dominanceVar is used in place of numVar. Specifying dominanceVar is beneficial for avoiding warnings when there are multiple numVar variables. Typically, dominanceVar will be one of the variables already included in numVar. |

## Details

This function marks input cells as singletons according to ones in data[[nUniqueVar]], if available, and otherwise according to data[[freqVar]]. The output vector can be logical or integer. When, integer, singletons are given as positive values. Their unique values represent the unique values/combinations of data[[charVar]].

## Value

logical or integer vector

## Note

SingletonUniqueContributor0 is a special version that produces singleton as a two-element list. See GaussSuppression and SuppressDominantCells.

## Examples

```
S <- function(data, ...) {
  cbind(data, singleton = SingletonUniqueContributor(data, ...))
}
d2 <- SSBtoolsData("d2")
d <- d2[d2$freq < 5, ]
d$nUnique <- round((5 - d$freq)/3)
d$freq <- round(d$freq/2)
d[7:8, 2:4] <- NA
rownames(d) <- NULL

S(d, freqVar = "freq", integerSingleton = FALSE)
S(d, freqVar = "freq", nUniqueVar = "nUnique", integerSingleton = TRUE, charVar = "main_income")
S(d, nUniqueVar = "nUnique", integerSingleton = TRUE, charVar = c("main_income", "k_group"))
S(d, freqVar = "freq", nUniqueVar = "nUnique", integerSingleton = FALSE,
  charVar = "main_income", removeCodes = "other")
S(d, nUniqueVar = "nUnique", integerSingleton = FALSE, charVar = c("main_income", "k_group"),
  removeCodes = c("other", "400"))
S(d, nUniqueVar = "nUnique", integerSingleton = FALSE, charVar = c("main_income", "k_group"),
  removeCodes = data.frame(anyname = c("other", "400")))
```

```
S(d, nUniqueVar = "nUnique", integerSingleton = FALSE, charVar = c("main_income", "k_group"),
  removeCodes = list(main_income = c("other", "pensions"), k_group = "300"))
S(d, nUniqueVar = "nUnique", integerSingleton = FALSE, charVar = c("main_income", "k_group"),
  removeCodes = data.frame(main_income = "other", k_group = "400"))
S(d, nUniqueVar = "nUnique", integerSingleton = FALSE, removeCodes = 1:5)

x <- SSBtools::ModelMatrix(d, hierarchies = list(region = "Total"))
which(colSums(x) == 1)
which(rowSums(x[, colSums(x) == 1]) > 0)
# columns 2, 3, 4, 5, 7 correspond to inner cells: rows 3, 4, 5, 6, 8
# with 2:4 not primary rows 3:5 are forced non-singleton
S(d, freqVar = "freq", nUniqueVar = "nUnique", integerSingleton = FALSE, x = x, primary = 5:8)
```

---

`SuppressDirectDisclosure` *Suppression of directly-disclosive cells*

---

## Description

Function for suppressing directly-disclosive cells in frequency tables. The method detects and primary suppresses directly-disclosive cells with the FindDisclosiveCells function, and applies a secondary suppression using Gauss suppression (see GaussSuppressionFromData).

## Usage

```
SuppressDirectDisclosure(
  data,
  dimVar,
  freqVar,
  coalition = 1,
  secondaryZeros = coalition,
  candidates = DirectDisclosureCandidates,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | the input data |
| `dimVar` | main dimensional variables for the output table |
| `freqVar` | variable containing frequency counts |
| `coalition` | numeric variable, parameter for primary suppression. Default value is 1. |
| `secondaryZeros` | logical or numeric value for secondary suppression. If logical, it is converted to resp numeric value (0 or 1). If numeric, it describes the largest number that is prioritized over zeroes in secondary suppression. Default value is equal to coalition. |
| `candidates` | function parameter for gauss suppression. |

... optional parameters that can be passed to the primary suppression method. See FindDisclosiveCells for details. In the case of SuppressDirectDisclosure2, ... are parameters to GaussSuppressionFromData.

## Details

SuppressDirectDisclosure has no support for hierarchical data. SuppressDirectDisclosure2 has, but is less general in other ways.

## Value

data.frame containing the result of the suppression

## Author(s)

Daniel Lupp

## Examples

```
tex <- data.frame(v1 = rep(c('a', 'b', 'c'), times = 4),
                  v2 = c('i','i', 'i','h','h','h','i','i','i','h','h','h'),
                  v3 = c('y', 'y', 'y', 'y', 'y', 'y','z','z', 'z', 'z', 'z', 'z'),
                  freq = c(0,0,5,0,2,3,1,0,3,1,1,2))
SuppressDirectDisclosure(tex, c("v1", "v2", "v3"), "freq")
SuppressDirectDisclosure(tex, c("v1", "v2", "v3"), "freq", coalition = 2, unknown.threshold = 10)

z3 <- SSBtools::SSBtoolsData("z3")
a1 <- SuppressDirectDisclosure(z3, c(1, 4, 5), 7)
b1 <- try(SuppressDirectDisclosure(z3, 1:6, 7))
```

---

# SuppressDominantCells *Suppress magnitude tables using dominance* (n,k) *or p% rule for primary suppression.*

---

## Description

Suppress magnitude tables using dominance (n,k) or p% rule for primary suppression.

## Usage

```
SuppressDominantCells(
  data,
  n = 1:length(k),
  k = NULL,
  pPercent = NULL,
  allDominance = FALSE,
  dominanceVar = NULL,
  numVar = NULL,
```

```
  dimVar = NULL,
  hierarchies = NULL,
  formula = NULL,
  contributorVar = NULL,
  sWeightVar = NULL,
  ...,
  candidatesVar = NULL,
  singletonZeros = FALSE,
  spec = PackageSpecs("dominanceSpec")
)
```

## Arguments

| | |
|---|---|
| data | Input data as a data frame |
| n | Parameter n in dominance rule. Default is 1:length(k). |
| k | Parameter k in dominance rule. |
| pPercent | Parameter in the p% rule, when non-NULL. Parameters n and k will then be ignored. Technically, calculations are performed internally as if n = 1:2. The results of these intermediate calculations can be viewed by setting allDominance = TRUE. |
| allDominance | Logical parameter. If TRUE, adds primary columns for each pair of parameters n,k in the dominance rules |
| dominanceVar | Numerical variable to be used in dominance rule. The first numVar variable will be used if it is not specified. |
| numVar | Numerical variable to be aggregated. Any dominanceVar and candidatesVar that are specified and not included in numVar will be aggregated accordingly. |
| dimVar | The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified. |
| hierarchies | List of hierarchies, which can be converted by AutoHierarchies. Thus, the variables can also be coded by "rowFactor" or "", which correspond to using the categories in the data. |
| formula | A model formula |
| contributorVar | Extra variables to be used as grouping elements in the dominance rule. Typically, the variable contains the contributor IDs. |
| sWeightVar | Name of variable which represents sampling weights to be used in dominance rule |
| ... | Further arguments to be passed to the supplied functions and to ModelMatrix (such as inputInOutput and removeEmpty). |
| candidatesVar | Variable to be used in the candidate function to prioritize cells for publication and thus not suppression. If not specified, the same variable that is used for the dominance rule will be applied (see dominanceVar and numVar). |
| singletonZeros | When negative values cannot occur, one can determine from a non-suppressed marginal cell with the value 0 that all underlying cells also have the value 0. The use of singletonZeros = TRUE is intended to prevent this phenomenon from |

causing suppressed cells to be revealable. It is the zeros in the dominanceVar variable that are examined. Specifically, the ordinary singleton method is combined with a method that is actually designed for frequency tables. This approach also works for volume tables when SingletonUniqueContributor0 is utilized.

spec NULL or a named list of arguments that will act as default values.

## Value

data frame containing aggregated data and suppression information.

## Examples

```
num <- c(100,
         90, 10,
         80, 20,
         70, 30,
         50, 25, 25,
         40, 20, 20, 20,
         25, 25, 25, 25)
v1 <- c("v1",
        rep(c("v2", "v3", "v4"), each = 2),
        rep("v5", 3),
        rep(c("v6", "v7"), each = 4))
sweight <- c(1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1)
d <- data.frame(v1 = v1, num = num, sweight = sweight)

# basic use
SuppressDominantCells(d, n = c(1,2), k = c(80,70), numVar = "num", formula = ~v1 -1)
SuppressDominantCells(d, k = c(80,70), numVar = "num", formula = ~v1 -1) # same as above
SuppressDominantCells(d, pPercent = 7, numVar = "num", formula = ~v1 -1)

# with weights
SuppressDominantCells(d, n = c(1,2), k = c(80,70), numVar = "num",
dimVar = "v1", sWeightVar = "sweight")

# overwriting some parameters in default spec
SuppressDominantCells(d, n = c(1,2), k = c(80,70), numVar = "num",
dimVar = "v1", sWeightVar = "sweight", domWeightMethod = "tauargus")

# using dominance and few contributors rule together, see second example compared to first
SuppressDominantCells(d, n = c(1,2), k = c(80,70), numVar = "num", formula = ~v1 -1,
primary = c(DominanceRule, NContributorsRule), maxN = 3, allDominance = TRUE)

SuppressDominantCells(d, n = c(1,2), k = c(80,70), numVar = "num", formula = ~v1 -1,
primary = c(DominanceRule, NContributorsRule), maxN = 4, allDominance = TRUE)


d2 <- SSBtoolsData("d2")
set.seed(123)
d2$v <- rnorm(nrow(d2))^2
```

```
# Hierarchical region variables are detected automatically -> same output column
SuppressDominantCells(data = d2, n = c(1, 2), k = c(70, 95), numVar = "v",
                      dimVar = c("region", "county", "k_group"), allDominance = TRUE)

# Formula. Hierarchical variables still detected automatically.
SuppressDominantCells(data = d2, n = c(1, 2), k = c(70, 95), numVar = "v",
                      formula = ~main_income * k_group + region + county - k_group)

# With hierarchies created manually
ml <- data.frame(levels = c("@", "@@", "@@@", "@@@", "@@@", "@@"),
       codes = c("Total", "not_assistance", "other", "pensions", "wages", "assistance"))
SuppressDominantCells(data = d2, n = c(1, 2), k = c(70, 95), numVar = "v",
                      hierarchies = list(main_income = ml, k_group = "Total_Norway"))
```

---

`SuppressFewContributors` *Few contributors suppression*

---

## Description

This function provides functionality for suppressing volume tables based on the few contributors rule (NContributorsRule).

## Usage

```
SuppressFewContributors(
  data,
  maxN,
  numVar = NULL,
  dimVar = NULL,
  hierarchies = NULL,
  formula = NULL,
  contributorVar = NULL,
  removeCodes = character(0),
  remove0 = TRUE,
  candidatesVar = NULL,
  ...,
  spec = PackageSpecs("fewContributorsSpec")
)
```

## Arguments

| | |
|---|---|
| `data` | Input data as a data frame |
| `maxN` | Suppression parameter. Cells with frequency <= `maxN` are set as primary suppressed. Using the default `primary` function, `maxN` is by default set to `3`. See details. |

| | |
|---|---|
| numVar | Numerical variable to be aggregated. Any candidatesVar that is specified and not included in numVar will be aggregated accordingly. Additionally, if remove0 is specified as a variable name and it is not included in numVar, it will also be aggregated accordingly. See parameters candidatesVar and remove0 below. |
| dimVar | The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified. |
| hierarchies | List of hierarchies, which can be converted by AutoHierarchies. Thus, the variables can also be coded by "rowFactor" or "", which correspond to using the categories in the data. |
| formula | A model formula |
| contributorVar | Extra variables to be used as grouping elements when counting contributors. Typically, the variable contains the contributor IDs. |
| removeCodes | Vector of codes to be omitted when counting contributors. With empty contributorVar row indices are assumed and conversion to integer is performed. |
| remove0 | When set to TRUE (default), data rows in which the first numVar (if any) is zero are excluded from the count of contributors. Alternatively, remove0 can be specified as one or more variable names. In this case, all data rows with a zero in any of the specified variables are omitted from the contributor count. Specifying remove0 as variable name(s) is useful for avoiding warning when there are multiple numVar variables. |
| candidatesVar | Variable to be used in the candidate function to prioritize cells for publication and thus not suppression. The first numVar variable will be used if it is not specified. |
| ... | Further arguments to be passed to the supplied functions and to ModelMatrix (such as inputInOutput and removeEmpty). |
| spec | NULL or a named list of arguments that will act as default values. |

## Value

data.frame containing aggregated data and supppression information. Columns nRule and nAll contain the number of contributors. In the former, removeCodes is taken into account.

## Examples

```
num <- c(100,
         90, 10,
         80, 20,
         70, 30,
         50, 25, 25,
         40, 20, 20, 20,
         25, 25, 25, 25)
v1 <- c("v1",
        rep(c("v2", "v3", "v4"), each = 2),
        rep("v5", 3),
        rep(c("v6", "v7"), each = 4))
sweight <- c(1, 2, 1, 2, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1)
d <- data.frame(v1 = v1, num = num, sweight = sweight)
```

```
SuppressFewContributors(d, formula = ~v1, maxN = 1, numVar = "num")
SuppressFewContributors(d, formula = ~v1, maxN = 2, numVar = "num")
SuppressFewContributors(d, formula = ~v1, maxN = 3, numVar = "num")


d2 <- SSBtoolsData("d2")[-5]
set.seed(123)
d2$v <- round(rnorm(nrow(d2))^2, 1)
d2$family_id <- round(2*as.integer(factor(d2$region)) + runif(nrow(d2)))

# Hierarchical region variables are detected automatically -> same output column
SuppressFewContributors(data = d2, maxN = 2, numVar = "v", contributorVar = "family_id",
                        dimVar = c("region", "county", "k_group"))

# Formula. Hierarchical variables still detected automatically.
# And codes 1:9 not counted
SuppressFewContributors(data = d2, maxN = 1, numVar = "v", contributorVar = "family_id",
                        formula = ~main_income * k_group + region + county - k_group,
                        removeCodes = 1:9)

# With hierarchies created manually
ml <- data.frame(levels = c("@", "@@", "@@@", "@@@", "@@@", "@@"),
       codes = c("Total", "not_assistance", "other", "pensions", "wages", "assistance"))
SuppressFewContributors(data = d2, maxN = 2, numVar = "v", contributorVar = "family_id",
                        hierarchies = list(main_income = ml, k_group = "Total_Norway"))
```

---

SuppressionFromDecimals

*Cell suppression from synthetic decimal numbers*

---

## Description

Decimal numbers, as calculated by GaussSuppressDec, are used to decide suppression (whole numbers or not). Technically, the calculations are done via GaussSuppressionFromData, but without running GaussSuppression. All suppressed cells are primary suppressed.

## Usage

```
SuppressionFromDecimals(
  data,
  decVar,
  freqVar = NULL,
  numVar = NULL,
  preAggregate = FALSE,
  digits = 9,
  ...
)
```

### Arguments

| | |
|---|---|
| data | Input data as a data frame |
| decVar | One ore several (nRep>1) decimal number variables. |
| freqVar | A single variable holding counts (not needed) |
| numVar | Other numerical variables to be aggregated |
| preAggregate | Parameter to GaussSuppressionFromData |
| digits | Parameter to RoundWhole. Values close to whole numbers will be rounded. |
| ... | Other parameters to GaussSuppressionFromData |

### Details

Several decimal number variables reduce the probability of obtaining whole numbers by chance.

### Value

Aggregated data with suppression information

### Author(s)

Øyvind Langsrud

### Examples

```
z2 <- SSBtoolsData("z2")

# Find suppression and decimal numbers with "fylke" in model
a <- GaussSuppressDec(z2, dimVar = c("region", "fylke", "hovedint"),
                      freqVar = "ant", protectZeros = FALSE, maxN = 2,
                      output = "inner")

# Add decimal numbers to data
z2$freqDec <- a$freqDec

# Find suppression with "kostragr" in model
b <- SuppressionFromDecimals(z2, dimVar = c("region", "kostragr", "hovedint"),
                             freqVar = "ant", decVar = "freqDec")
```

---

## SuppressKDisclosure *K-disclosure suppression*

---

### Description

A function for suppressing frequency tables using the k-disclosure method.

## Usage

```
SuppressKDisclosure(
  data,
  coalition = 0,
  mc_hierarchies = NULL,
  upper_bound = Inf,
  dimVar = NULL,
  formula = NULL,
  hierarchies = NULL,
  freqVar = NULL,
  ...,
  spec = PackageSpecs("kDisclosureSpec")
)
```

## Arguments

| | |
|---|---|
| `data` | a data.frame representing the data set |
| `coalition` | numeric vector of length one, representing possible size of an attacking coalition. This parameter corresponds to the parameter k in the definition of k-disclosure. |
| `mc_hierarchies` | a hierarchy representing meaningful combinations to be protected. Default value is `NULL`. |
| `upper_bound` | numeric value representing minimum count considered safe. Default set to `Inf` |
| `dimVar` | The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified. |
| `formula` | A model formula |
| `hierarchies` | List of hierarchies, which can be converted by `AutoHierarchies`. Thus, the variables can also be coded by `"rowFactor"` or `""`, which correspond to using the categories in the data. |
| `freqVar` | name of the frequency variable in `data` |
| `...` | parameters passed to children functions |
| `spec` | `NULL` or a named list of arguments that will act as default values. |

## Value

A data.frame containing the publishable data set, with a boolean variable `$suppressed` representing cell suppressions.

## Author(s)

Daniel P. Lupp

## Examples

```
# data
data <- SSBtools::SSBtoolsData("mun_accidents")

# hierarchies as DimLists
mun <- data.frame(levels = c("@", rep("@@", 6)),
codes = c("Total", paste("k", 1:6, sep = "")))
inj <- data.frame(levels = c("@", "@@" ,"@@", "@@", "@@"),
codes = c("Total", "serious", "light", "none", "unknown"))
dimlists <- list(mun = mun, inj = inj)

inj2 <- data.frame(levels = c("@", "@@", "@@@" ,"@@@", "@@", "@@"),
codes = c("Total", "injured", "serious", "light", "none", "unknown"))
inj3 <- data.frame(levels = c("@", "@@", "@@" ,"@@", "@@"),
codes = c( "shadowtotal", "serious", "light", "none", "unknown"))
mc_dimlist <- list(inj = inj2)
mc_nomargs <- list(inj = inj3)

#' # Example with formula, no meaningful combination
out <- SuppressKDisclosure(data, coalition = 1, freqVar = "freq", formula = ~mun*inj)

# Example with hierarchy and meaningful combination
out2 <- SuppressKDisclosure(data, coalition = 1, freqVar = "freq",
hierarchies = dimlists, mc_hierarchies = mc_dimlist)

#' # Example of table without mariginals, and mc_hierarchies to protect
out3 <- SuppressKDisclosure(data, coalition = 1, freqVar = "freq",
formula = ~mun:inj, mc_hierarchies = mc_nomargs )
```

---

`SuppressSmallCounts` *Small count frequency table suppression.*

---

## Description

This is a wrapper function of `GaussSuppressionFromData` for small count frequency suppression. For common applications, the spec parameter can be adjusted, see `PackageSpecs` for more information. See Details for more information on function call customization.

## Usage

```
SuppressSmallCounts(
  data,
  maxN,
  freqVar = NULL,
  dimVar = NULL,
  hierarchies = NULL,
  formula = NULL,
  ...,
  spec = PackageSpecs("smallCountSpec")
)
```

## Arguments

| | |
|---|---|
| data | Input data as a data frame |
| maxN | Suppression parameter. Cells with frequency <= maxN are set as primary suppressed. Using the default primary function, maxN is by default set to 3. See details. |
| freqVar | A single variable holding counts (name or number). |
| dimVar | The main dimensional variables and additional aggregating variables. This parameter can be useful when hierarchies and formula are unspecified. |
| hierarchies | List of hierarchies, which can be converted by AutoHierarchies. Thus, the variables can also be coded by "rowFactor" or "", which correspond to using the categories in the data. |
| formula | A model formula |
| ... | Further arguments to be passed to the supplied functions and to ModelMatrix (such as inputInOutput and removeEmpty). |
| spec | NULL or a named list of arguments that will act as default values. |

## Details

The specs provided in the package (see PackageSpecs) provide common parameter setups for small count suppression. However, it might be necessary to customize the parameters further. In this case, certain parameters from GaussSuppressionFromData might need adjusting from the values provided by the package specs. In particular, the parameters protectZeros (should zeros be primary suppressed), extend0 (should empty cells be added before primary suppression), and secondaryZeros (should zero frequency cells be candidates for secondary suppression) might be of interest. The examples below illustrate how to override parameters specified by a spec. Note that this is only possible if specLock = FALSE.

## Value

data frame containing aggregated data and suppression information.

## Examples

```
mun_accidents <- SSBtoolsData("mun_accidents")

SuppressSmallCounts(data = mun_accidents, maxN = 3, dimVar = 1:2, freqVar = 3)
# override default spec
SuppressSmallCounts(data = mun_accidents, maxN = 3, dimVar = 1:2, freqVar = 3,
                    protectZeros = FALSE)


d2 <- SSBtoolsData("d2")
d2$f <- round(d2$freq/10)  # tenth as frequency in examples

# Hierarchical region variables are detected automatically -> same output column
SuppressSmallCounts(data = d2, maxN = 2, freqVar = "f",
                    dimVar = c("region", "county", "k_group"))
```

```
# Formula. Hierarchical variables still detected automatically.
SuppressSmallCounts(data = d2, maxN = 3, freqVar = "f",
                    formula = ~main_income * k_group + region + county - k_group)

# With hierarchies created manually
ml <- data.frame(levels = c("@", "@@", "@@@", "@@@", "@@@", "@@"),
       codes = c("Total", "not_assistance", "other", "pensions", "wages", "assistance"))
SuppressSmallCounts(data = d2, maxN = 2, freqVar = "f",
                    hierarchies = list(main_income = ml, k_group = "Total_Norway"))


# Data without pensions in k_group 400
# And assume these are structural zeros (will not be suppressed)
SuppressSmallCounts(data = d2[1:41, ], maxN = 3, freqVar = "f",
                    hierarchies = list(main_income = ml, k_group = "Total_Norway"),
                    extend0 = FALSE, structuralEmpty = TRUE)
# -- Note for the example above --
# With protectZeros = FALSE
#   - No zeros suppressed
# With extend0 = FALSE and structuralEmpty = FALSE
#   - Primary suppression without protection (with warning)
# With extend0 = TRUE and structuralEmpty = TRUE
#   - As default behavior. Suppression/protection of all zeros (since nothing empty)
# With formula instead of hierarchies: Extra parameter needed when extend0 = FALSE.
#   - removeEmpty = FALSE,  to include empty zeros in output.
```

# Index