Protect 5-dimensional tables with the modular approach implemented in τ-Argus

3rd Workshop on Time Series Analysis and Statistical Disclosure Control Methods for Official Statistics 14-15

J. Jamme
W. Pomel
A.-R. Socard

2023/12/15

Measuring, understanding

École nationale de la statistique et de l'analyse de l'information

Context

- Protection of tabular data against disclosure risks
- SDC method: suppressive method applied on tabular data in two steps:
  - **Primary suppression** (frequency, dominance or p% rules) - easy step to compute
  - **Secondary suppression** (marginal differenciation, protection levels, singleton rules, etc.) - need of powerful algorithm to compute
- Reference tool: τ-Argus (CBS, [De Wolf et al., 2014]) - for the secondary suppression algorithms
- Additional tool: `rtauargus` (Insee, [Berrard et al., 2023]) - management of the suppression over large sets of linked tables

Context

There are 4 secondary suppression methods implemented in τ-Argus . Especially,

- `Hypercube` (GHMITER, [Repsilber, 1994])
- `Modular` (`HiTas`, [De Wolf, 2002])

- *Advantages*:
  - Fastest method in τ-Argus
  - Can deal with long and large tables.
- *Drawback*:
  - Many suppressed cells,
  - Margins are easily suppressed,
  - $\Rightarrow$ Low utility.

- *Advantages*:
  - Fast (but less than `Hypercube`),
  - Good utility: margins cells are more protected than the inner cells (Minimisation of the suppressed values)
- *Drawbacks*:
  - In some cases, problems are infeasible for `Modular` (lots of zeroes in sub-tables [De Wolf et al., 2014]),
  - Very long computation time on some tables (3 days for a $2D$ table with 360 000 rows),
  - `Modular` is available only for $\leq 4D$ tables.

$\Longrightarrow$ `Modular` is the best way to solve secondary suppression problems.

Motivations

- Need to release some large tabular data (> 4 dims)
- Today, only possible with `Hypercube` $\Rightarrow$ Very low utility

$\Rightarrow$ Current alternative at Insee is:

- Either apply home made algorithm but with no optimisation, no protection levels, no relevancy between linked tables
- Or not release the $5D$ tabular data but only some $3D$ or $4D$ tables.

The mission

What if we could find a way to reduce the number of dimensions of tabular data without loosing cells ?

Issues/Challenges

- How to do this ?
- Does the solution generate less suppression than `Hypercube` ?
- Is the solution adapted to our real use cases ?

An example to start

Let's have a $4D$ tabular data crossing:

- $EDU \in \{ALL, A, B, C, D\}$
- $OCC \in \{ALL, A, B, C\}$
- $GEN \in \{ALL, F, M\}$
- $AGE \in \{ALL, Child, Adult\}$

An example to start

EDU	OCC	GEN	AGE	FREQ
A	A	ALL	ALL	86
A	A	ALL	Adult	44
A	A	ALL	Child	42
A	A	F	ALL	36
A	A	F	Adult	23
A	A	F	Child	13
A	A	M	ALL	50
A	A	M	Adult	21
A	A	M	Child	29

Table: First nine rows (over 180) of the original 4-dimensional table to split

An example to start

Naive idea: merge two variables

EDU	OCC	GEN_AGE	FREQ
A	A	ALL_ALL	86
A	A	ALL_Adult	44
A	A	ALL_Child	42
A	A	F_ALL	36
A	A	F_Adult	23
A	A	F_Child	13
A	A	M_ALL	50
A	A	M_Adult	21
A	A	M_Child	29

Table: First nine rows (over 180) of the 3-dimensional table after first step (merging step)

An example to start

Naive idea ⇒ non-nested hierarchies issue

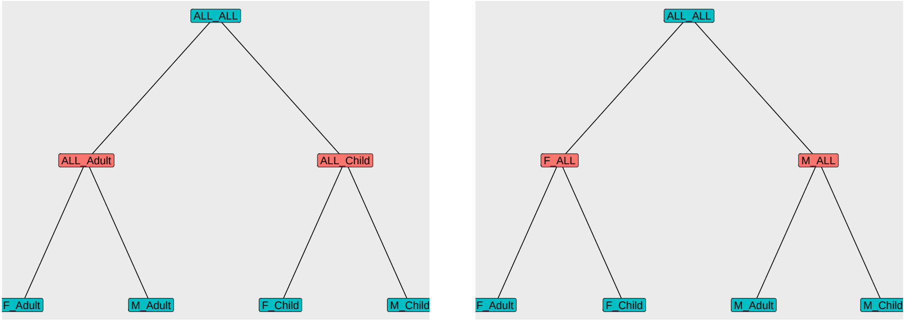

Figure: The two non-nested hierarchies after the merging of GENDER and AGE variables

An example to start

Idea: Merge-and-split the table

EDU	OCC	GEN_AGE	FREQ
A	A	ALL_ALL	86
A	A	ALL_Adult	44
A	A	ALL_Child	42
A	A	F_Adult	23
A	A	F_Child	13
A	A	M_Adult	21
A	A	M_Child	29

EDU	OCC	GEN_AGE	FREQ
A	A	ALL_ALL	86
A	A	F_ALL	36
A	A	F_Adult	23
A	A	F_Child	13
A	A	M_ALL	50
A	A	M_Adult	21
A	A	M_Child	29

Table: The two linked sub-tables to protect

- Merge step: Remove one dimension by merging two of the original variables
- Split step:
  - Split the table in sub-tables
  - In each sub-table, the merged variable has to be perfectly hierarchical

How to split the table in any cases?

$\Longleftrightarrow$

How to detect all the non-nested hierarchies in the new variable?

Notations:

- X, Y variables to merge (*ie* paste the categories)
- X_Y the merged variable
- n_X, n_Y, number of nodes of X, Y respectively,
- The **nodes** are all the categories of a hierarchy except the leaves,
- A **sub-part of a hierarchy** consists of a node and the categories immediately below it.

There are as many sub-parts in a hierarchy as nodes

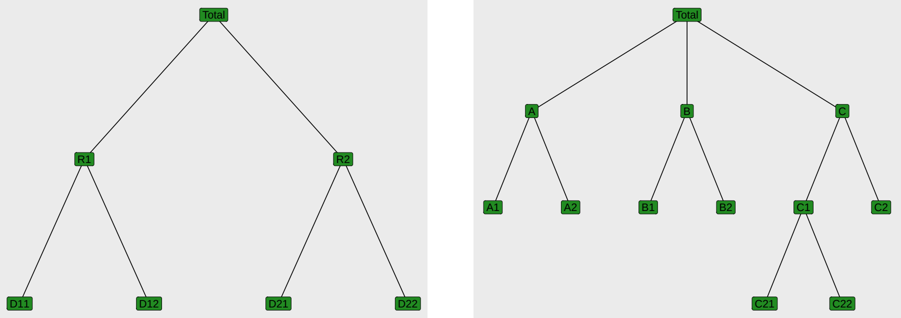

FIGURE: Example of hierarchies for X (left) and Y (right)

- $n_X = 3$ and $n_Y = 5$ nodes
- $\{Total, R1, R2\}$ is a sub-part of X's hierarchy from the node $Total$
- $\{B, B1, B2\}$ is a sub-part of Y's hierarchy from the node B

First Step: Build sub-tables from the hierarchy of X

- One sub-table for each sub-part of the X's hierarchy
- Keep all the categories of Y
- $\Rightarrow$ We get n_X sub-tables

Example:

- T_1: $\cdots \times X\{Total, R1, R2\} \times Y$
- T_2: $\cdots \times X\{R1, D11, D12\} \times Y$
- T_3: $\cdots \times X\{R2, D21, D22\} \times Y$

Second Step: Build sub-tables from the hierarchy of Y

- One sub-table for each sub-part of the Y's hierarchy
- Apply this on each sub-table of the firs step
- $\Rightarrow$ We get $n_X \times n_Y$ sub-tables

Example:

- T_{11}: $\cdots \times X\{Total, R1, R2\} \times Y\{Total, A, B, C\}$
- T_{12}: $\cdots \times X\{Total, R1, R2\} \times Y\{A, A1, A2\}$
- T_{13}: $\cdots \times X\{Total, R1, R2\} \times Y\{B, B1, B2\}$
- T_{14}: $\cdots \times X\{Total, R1, R2\} \times Y\{C, C1, C2\}$
- T_{15}: $\cdots \times X\{Total, R1, R2\} \times Y\{C1, C21, C22\}$
- T_{21}: $\cdots \times X\{R1, D11, D12\} \times Y\{Total, A, B, C\}$
- $\vdots$
- T_{35}: $\cdots \times X\{R2, D21, D22\} \times Y\{C1, C21, C22\}$

Third Step: Merge the two variables

- To do in each sub-table
- $\Rightarrow$ We retrieve our starting example: X and Y have 1 node in each sub-table.

Fourth Step: Split each sub-table in two sub-tables to deal with non-nested hierarchies

- We get $2 \times n_X \times n_Y$ sub-tables
- X_Y is perfectly hierarchical in each one.

Here are the 30 sub-tables of the example to protect:

- T_{111}: $\cdots \times X\_Y\{Total\_Total, R1\_Total, R2\_Total, R1\_A, R1\_B, R1\_C\ R2\_A, R2\_B, R2\_C\}$
- T_{112}: $\cdots \times X\_Y\{Total\_Total, Total\_A, Total\_B, Total\_C, R1\_A, R2\_A, R1\_B, R2\_B, R1\_C\ R2\_C\}$
- $\vdots$
- T_{351}: $\cdots \times X\_Y\{R2\_C1, D21\_C1, D22\_C1, D21\_C21, D21\_C22, D22\_C21, D22\_C22\}$
- T_{352}: $\cdots \times X\_Y\{R2\_C1, R2\_C21, R2\_C22, D21\_C21, D22\_C21, D21\_C22, D22\_C22\}$

WHAT ABOUT $5D$-TABLES

Principle: **Repeat the merge-and-split process twice consecutively**
There are two ways to do this:

- Either choose at both merge-and-split processes two different couples of variables ($\Rightarrow X\_Y$ and Z_T)
  - We get $(2 * n_X * n_Y) \times (2 * n_Z * n_T) = 4 * n_X * n_Y * n_Z * n_T$ sub-tables
  - Least number of sub-tables: $n_X = 1, n_Y = 1, n_Z = 1, n_T = 1 \Rightarrow 4$ sub-tables
- Or, at the second time, reuse the first merged variable ($\Rightarrow X\_Y\_Z$)
  - We get $12 * n_X * n_Y * n_Z$ sub-tables (for some cases, demo in the paper)
  - Least number of sub-tables: $n_X = 1, n_Y = 1, n_Z = 1 \Rightarrow 12$ sub-tables

The merge-and-split process produces a set of linked tables
$\Rightarrow$ We can expect over-suppression and longer computation time.

- Is the over-suppression acceptable ?
- How much longer is the computation ?
- Does the quantity of primary cells have an effect on over-suppression and computation time ?
- Is `Modular` combined with merge-and-split method better than `Hypercube` ?

Experimental Results

4D tables, no hierarchy

Results on 100 simulations:

method	Nb of linked tables	primary suppression n cells (%)	primary suppression values (%)	secondary suppression n cells (%)	secondary suppression values (%)	time comput.
1-Direct Modular	1	10.4	0.8	16.8	5.5	3.9
2-Split Modular	2	10.4	0.8	19.3	6.1	10.6
3-Direct Hypercube	1	10.4	0.8	34.6	13.8	3.6
4-Split Hypercube	2	10.4	0.8	21.1	6.6	11.4

Table: 4-Dimensional table - split on two non-hierarchical variables - 10% of primary cells

Experimental Results

4D tables, no hierarchy

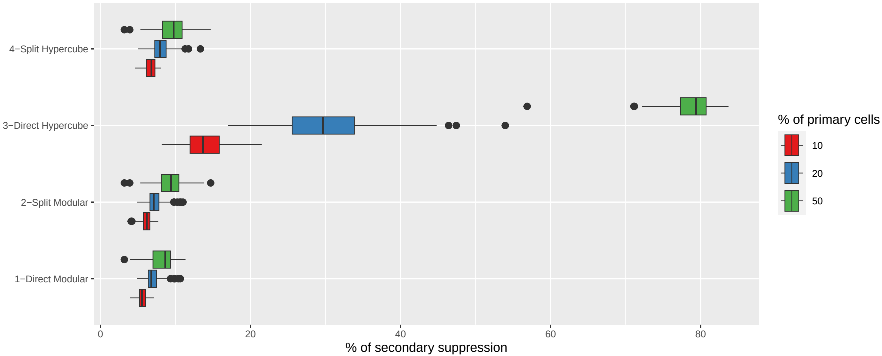

Figure: Distribution of % of values suppressed at the secondary step depending on the method and the % of primary cells for a 4-dimensional tabular data with no hierarchical variable.

method	Nb of linked tables	primary suppression n cells (%)	primary suppression values (%)	secondary suppression n cells (%)	secondary suppression values (%)	time comput.
2-Split Modular	4	20.2	2.7	20.6	7.8	27.7
3-Direct Hypercube	1	20.2	2.7	60.8	55.9	5.1
4-Split Hypercube	4	20.2	2.7	21.1	7.9	26.8

Table: Simulations on a 5-Dimensional table - split on two non-hierarchical variables - 20% of primary cells

Experimental Results

5D tables, no hierarchy

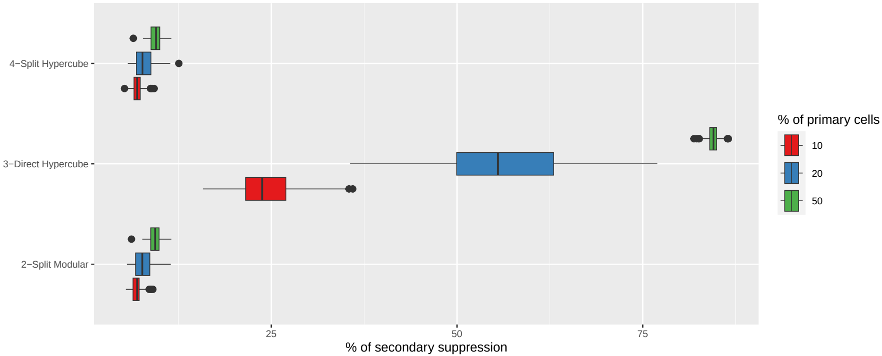

Figure: Distribution of % of values suppressed at the secondary step depending on the method and the % of primary cells for a 5-dimensional tabular data with no hierarchical variable.

- Is the over-suppression acceptable ? **Yes**
- How much longer is the computation ? 3 **times longer in the simulations**
- Does the quantity of primary cells have an effect on over-suppression and computation time ? **Yes, the variability is greater for a Split Modular than for a Direct Modular**
- Is `Modular` combined with merge-and-split method better than `Hypercube` ? **Always in terms of suppression, never in terms of time**

Discussion

To go further

Simulation results cannot be generalized to all real cases encountered:

- The efficiency of the secondary suppression is depending on the pattern of primary cells
- In real use cases, primary cells are not uniformly spread in the data that can produce some local difficulties for `Modular`
- Uniformly distributed primary cells seem to be a too gentle hypothesis $\Rightarrow$ Try more sophisticated distribution.

Discussion

To go further

- In some realistic examples of $4D$ table, split modular was faster than direct modular
  $\Rightarrow$ There are real use cases for which the merge-and-split method could be useful.
- On some real use cases ($5D$ table with 5 000 rows), split modular was very (too) long on each table.

Berrard, P.-Y., Jamme, J., Rastout, N., Beroud, F., Pointet, J., Pomel, W., and Socard, A.-R. (2023).
rtauargus: Run τ-Argus from R.
R package version 1.2.0 (dev).

De Wolf, P.-P. (2002).
Hitas: a heuristic approach to cell suppression in hierarchical tables.
In *Proceedings of the AMRADS meeting in Luxembourg*.

De Wolf, P.-P., Hundepool, A., Giessing, S., Salazar, J.-J., and Castro, J. (2014).
τ-Argus - User's Manual - Version 4.1.
Statistics Netherland.

Repsilber, R. D. (1994).
Preservation of confidentiality in aggregated data.
In *Second International Seminar on Statistical Confidentiality*.