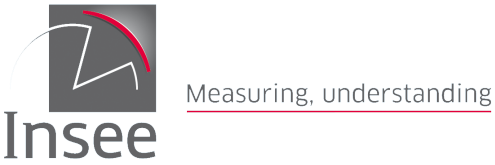
Protect 4 and 5-dimensional tables with the modular approach implemented in τ-Argus

Julien Jamme (Insee), Wistan Pomel (Ensai) and André-Raymond Socard (Ensai)

14-15 December 2023



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

Introduction

National Statistical Offices disseminate a lot of tabular data. Before releasing these tables, they have to protect the confidential information they can contain. A popular way to limit the disclosure is to use a suppressive approach: the sensitive cells are suppressed (primary step) and additional cells are also suppressed to make sure that the first ones can't be guessed by differencing with the margins (secondary step). The second step is, generally, done using either the `Hypercube` (`GHMITER`, [Repsilber, 1994]) method or the `Modular` (`HiTaS`, [De Wolf, 2002]) approach implemented in τ-Argus.

The `Hypercube` method is able to deal with large tables but the margins can easily be suppressed during the secondary suppression. On the contrary, the `Modular` approach tends to preserve the margins as much as possible but $\tau-$Argus prevents the use of `Modular` on 5-dimensional tables or larger. Then, while dealing with such large tables, we have, currently, only two options: apply hypercube or not release the table.

In this paper, we present a way to apply `Modular` on 5-dimensional tables with $\tau-$Argus. The idea is to split the original table into several sub-tables of lower dimensions, such that `Modular` can be used to apply secondary suppression on each sub-tables. To ensure the relevancy of the suppression pattern across all the sub-tables, we use the `tab_multi_manager()` function of the `rtauargus` package.

In the first section, the way to split a 4-dimensional table is explained, especially the way to detect all the non-nested hierarchies. In the second section, the process for 5-dimensional tables and its implementation in `rtauargus` are presented. We, then, present results of some simulations, and, finally, discuss the limits of this process.

1 The reduction of a 4-dimensional table

To reduce the number of dimensions of a 5-dimensional table and obtain 3-dimensional sub-tables, the reduction process needs to manage the reduction from a 4-dimensional table to 3-dimensional sub-tables. How do we proceed ?

Firstly, reducing from 4 to 3 dimensions involves merging two of the four dimensions of the original table, meaning two of the initial crossing variables are replaced by a single variable whose modalities (breakdowns) are the merging of the modalities of the two chosen variables. As this step creates non-nested hierarchies, it's necessary to split the table into several sub-tables to make sure that each table has a proper hierarchy on the new variable.

Then, we get a set of linked 3-dimensional tables on which the secondary suppression can be applied regardless of the number of tables in the set, thanks to the `rtauargus::tab_multi_manager` function which ensures that the suppressions are relevant across the set[1][Berrard et al., 2023]. Hence, the more complex step is the second one: split the tables so as to get tables with proper hierarchical variables.

1.1 A simple example to start

Let's have a 4-dimensional frequency table crossing the level of education (broken down in 4 categories), the occupation status (3 categories), the gender (2 categories) and the age of people (2 categories). A total category (denoted `ALL`) is added to the categories of each variable. We have then a table of $(4+1)*(3+1)*(2+1)*(2+1) = 180$ rows. The first rows of the table are displayed in table 1.

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

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

The first step consists in merging the categories of two variables. Here, we choose to merge the less broken down variables: `GENDER` and `AGE`. The resulting 3-dimensional table is displaying in table 2.

However, as shown in the figure 1, the modalities of the new variable are not nested in a single hierarchy. We get a problem of non-nested hierarchies. This can be solved by splitting the original table in as much hierarchies as we have to create such that the modalities in each hierarchy are well nested and that all the merged modalities are present in at least one of these hierarchies. In our simple example, only two hierarchies are necessary. So, we can split our table into two sub-tables (table 3). These two tables have common cells so they are linked together and the secondary suppression step has to take this into account.

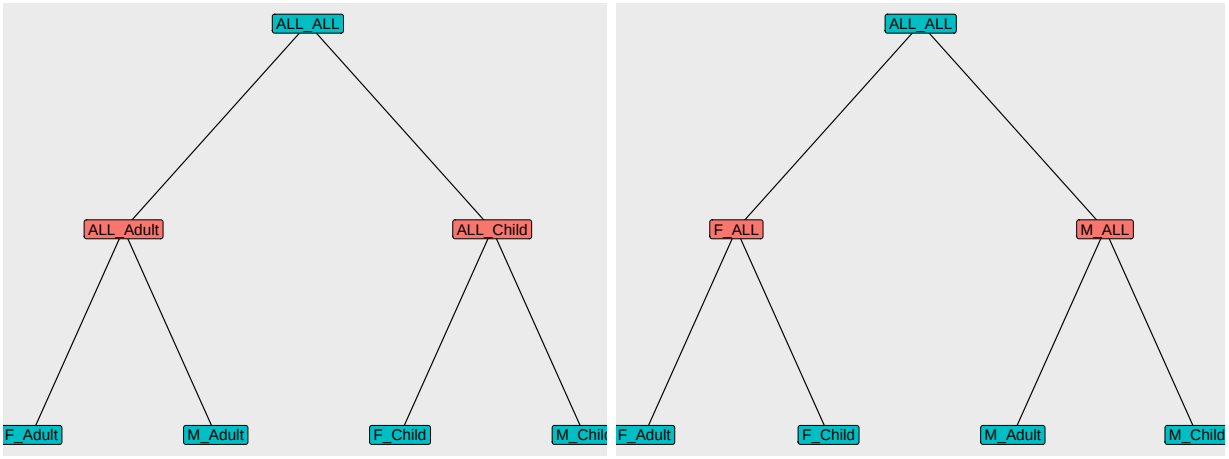


Figure 1: The two non-nested hierarchies after the merging of `GENDER` and `AGE` variables

[1]If the set contains not too many tables, $\tau-$Argus can deal with the linked tables directly.

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 3: The two linked sub-tables to protect

1.2 Detect all the non-nested hierarchies

The example is the simplest we can find. How can all the non-nested hierarchies be automatically detected? Actually, the number of sub-tables depends on the number of nodes within the hierarchies describing the nesting of the modalities of both variables to merge.

Let X and Y be the two variables to be merged. These variables each have possibly nested categories according to hierarchies containing n and m nodes respectively. The number of hierarchy nodes, is the number of subtotals in it, including the overall total of the hierarchy. While representing a hierarchy in the shape of a tree, the number of nodes of the tree is the number of nodes of the hierarchy.

For example, the hierarchy represented by the left tree (let's suppose it is X's hierarchy) in figure 2 has 3 nodes (`Total`, `R1` and `R2`), while, on the right (let's suppose it is Y's hierarchy), the tree has 5 nodes (`Total`, `A`, `B`, `C` and `C1`). $n = 1$ corresponds to the case when X has only an overall total without subtotals, as the `GENDER` variable used previously.

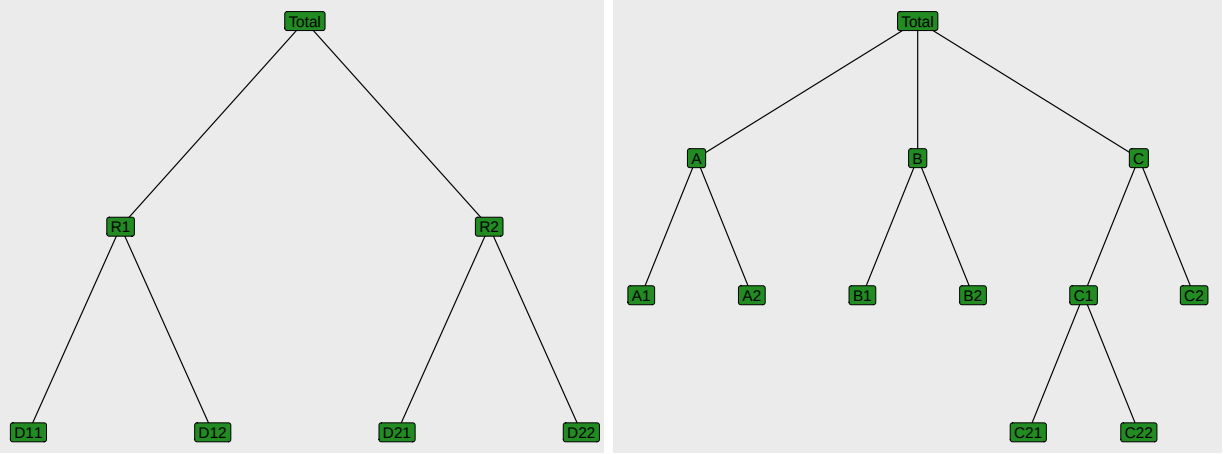


Figure 2: Two hierarchies with, respectively, 3 and 5 nodes

The first step involves constructing all possible sub-tables from one of the two variables to be merged, X for example. The idea is to filter the original table such that a sub-table is only composed of only one node of the X hierarchy and its leaves. This step generates as many tables as there are nodes in the X hierarchy (n).

From our example presented in the figure 2, this step results in the following tables if we start by working with the variable X:

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

The ... represents all other variables of the original table. Hence, T_1 crosses X restricted to the following categories `Total, R1, R2` with Y and all the other variables. So, X in T_1 is now a 1-node variable.

The second step involves performing the same operation on these sub-tables using the second variable to be merged (Y), that is to say that we produce, from each of these sub-tables as many tables as there are nodes in the hierarchy of Y.

In our example, the second variable to be merged is Y, which contains 5 nodes. Therefore, the second step consists in producing 5 sub-tables from each sub-table of the first step, so a total of 15 tables:

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

Hence, for each sub-table, we get two variables of 1-node hierarchy each. So, each table is as simple as the first example shown above (table 1). And we saw that such table generates two sub-tables while merging two of its variables to avoid non-nested hierarchies issue. So, a third step is necessary on our tables while merging X and Y variables: split all of them in two sub-tables. The first sub-table breaks down the overall total ($Total\_Total$ for the first sub-table) of the resulting variable X_Y by the categories of X ($R1\_Total$ and $R2\_Total$, for the first sub-table) and the second one uses the categories of Y ($Total\_A$, $Total\_B$ and $Total\_C$). Then, from our example, the resulting 30 sub-tables are the following ones:

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

All the hierarchies associated to the X_Y variable of the above 30 sub-tables are displayed in the appendix A.

1.3 How many sub-tables do we have to build ?

To remove one of the dimensions of a table, the number of sub-tables to build to avoid the non-nested hierarchies issue is $2 \times n \times m$, where n is the number of nodes in the hierarchy of the first merged variable and m the number of nodes in the hierarchy of the second one.

Indeed, at the first step, the number of sub-tables to create is equal to n since we create as many tables as there are nodes in the categories of X. At the second step, each of these n sub-tables is divided into as many tables as there are nodes in the categories of Y. Thus, we obtain $n \times m$ tables at the end of the second step. Finally, we merge the variables, and since this generates a double non-nested hierarchy, we divide each sub-table into two sub-tables. Therefore, we finally obtain $2 \times n \times m$ sub-tables.

From our first example, we had two variables with 1 node each and produced $2 * 1 * 1 = 2$ sub-tables. From our second example, we had two variables with, respectively, 3 and 5 nodes, and produced

$2 * 3 * 5 = 30$ sub-tables. Notice that the number of sub-tables can become very great if two highly broken down variables are used in the merging. For example, the NUTS3 $\subset$ NUTS2 $\subset$ NUST1 hierarchy contains 137 nodes. Even if the number of tables is not an issue while managing the suppression process of all the linked tables, the functions implemented in `rtauargus` can choose the variables which have the least number of nodes so as to produce the minimum of tables.

Notice that if both variables implied in the merging are non hierarchical (number of nodes equal to 1), then the reduction process builds only two sub-tables.

2 The reduction of a 5-dimensional table

To reduce a table from 5 to 3 dimensions, we use the previous method twice consecutively. Indeed, the method to reduce a 4-dimensional table to a set of 3-dimensional sub-tables is a general method to reduce a table into sub-tables with one less dimension. Then, the first step is to reduce the 5-dimensional table into a set of 4-dimensional sub-tables. And, the second step is to reduce each 4-dimensional table of the set into a set of 3-dimensional sub-tables.

As mentioned in the previous section, the choice of the variables to merge has some consequences on the number of the set of sub-tables. Here, the choice has to be made twice. For each step, the natural choice is to choose the two variables with the least number of nodes. Then, at the second reduction step, the merging can concerned either two other variables than those used at the first reduction step or the merged variable created at the first step is merged with another one. Then, there are two possibilities: either four variables $(2 + 2)$ are involved in the process during the reduction process or only three $(2 + 1)$.

Let's note X, Y, Z, T four of the five dimensions of our table, having, respectively, n_X, n_Y, n_Z, n_T nodes. Let's consider that at the first step, the process merges X and Y and at the second step, it merges Z and T. Then, the number of sub-tables at the end of the process is $(2{*}n_X{*}n_Y){\times}(2{*}n_Z{*}n_T) = 4 * n_X * n_Y * n_Z * n_T$.

The least number of nodes is always provided by the non-hierarchical variables, because they have only one node (the total category). Hence, choosing, if any, two couples of non hierarchical variables[2] among the five original variables will produce only four 3-dimensional tables.

If, only three variables are involved in the reduction process (X and Y at the first step and X_Y and Z at the second step, for example), the number of sub-tables is depending on the number of leaves of each nodes of the X and Y variables. A general formulation is complicated to display and needs complicated notations. But in the case when each node of the X and Y variables has only two leaves, we can show that $12 * n_X * n_Y * n_Z$ sub-tables are generated by our method. Indeed method, the variable X_Y created at the first step of reduction has exactly 3 nodes in each sub-table, in this particular case. In addition, at the second step, one sub-table is split into $2 * n_Z * n_{X\_Y} = 6 * n_Z$. *QED.*

2.1 Implementation in the `rtauargus` package

This reduction of dimensions of 4 and 5-dimensional tables is implemented in the `tab_rtauargus` and `tab_multi_manager` functions. The user has to set the `split_tab` argument to `TRUE` to proceed. Set to `FALSE`, the suppression process will be accompanied by a warning from τ-Argus for 4-dimensional tables and it won't be possible for 5-dimensional tables. `tab_rtauargus` is suited to protect a single tabular data. With `split_tab=TRUE` argument (default), if the table has 4 or 5 dimensions, the function automatically splits the original table into a set of 3-dimensional sub-tables and call the `tab_multi_manager` function to ensure the relevancy of the secondary suppression on all the tables of the set. `tab_multi_manager` is suited to protect a set of tabular data. With `split_tab=TRUE` argument (default), the function splits each 4 or 5-dimensional table of the set. At the end of the process, the original table(s) - with its original dimensions - is (are) retrieved, augmented with information about the secondary suppression. To use this functionality, it is necessary for the user to set the primary status of the cells him or herself beforehand.

During our experimentation, we noticed that, when the original 5-dimensional table is long (several tens of thousands of rows), the choice of splitting with the variables that have the least number of nodes (the non hierarchical ones in priority) leads to produce sub-tables that can remain quite long and

[2] one couple for each reduction step

that need long computation time in a `Modular` approach. That's why the `rtauargus` implementation includes an option to reduce the size of the sub-tables in choosing hierarchical variables instead. This leads to increase the number of sub-tables and reduce the length of the tables. Unfortunately, this option can't solve all issues encountered when dealing with big tables.

3 Experimental results

3.1 Direct vs *merge-and-split* approaches on 4-dimensional tables without hierarchies

As τ-Argus is able to deal with 4-dimensional tables with `Modular`, we can compare secondary suppression and computation time between a direct approach (*ie* let $\tau-$Argus do the job on the original table) and a *merge-and-split* approach using our merge-and-split process. We also compare `Hypercube` and `Modular` methods.

In the first three sets of simulations whose results are displayed in the three tables 4, 5 and 6, the original data is a 4-dimensional table with non-hierarchical crossing variables made up of 5, 5, 3 and 3 modalities (including the total). The original data has 225 rows. As there is no hierarchical variables, the dimension reduction method generates only two linked tables.

The frequencies of the inner cells of the table are generated with a uniform distribution on $[0; 100]$ and rounded to the superior integer. Hence, there is no zeros in the data. It's not realistic but more convenient for the simulations, because the presence of zeros can lead to infeasible solution. For each set of simulations, 100 tables have been generated. A frequency rule is applied to detect the primary cells on each table. The threshold is defined as the quantile of the frequency variable corresponding to the percentage of primary secret we want to introduce. table 4 concerns simulations with 10% of primary cells introduced in the tabular data, table 5 corresponds to 20% of primary suppression and table 6 50% of primary cells have been created. Once the primary secret is set, on each simulated tabular data, four different suppression methods have been applied:

- `1-Direct Modular`: `Modular` is directly applied on the original 4-dimensional table.
- `2-Split Modular`: `Modular` is applied on the set of 3-dimensional sub-tables created by our method of reduction of the dimensions.
- `3-Direct Hypercube`: `Hypercube` is directly applied on the original 4-dimensional table.
- `4-Split Hypercube`: `Hypercube` is applied on the set of 3-dimensional sub-tables created by our method of reduction of the dimensions.

In each table of results below, the following statistics are displayed:

- `Nb of linked tables` is the number of linked tables on which the secondary suppression has been applied. It's always 1 for *direct* approach and 2 for *merge-and-split* approach.
- `primary suppression (n and values)` columns are the percentage of primary cells and the percentage of values of them. The values don't depend on the method. Then, they are identical for all of them. The percentage of primary cells displayed is the mean over the 100 simulations. The value is reported to check that the target is reached.
- `secondary suppression (n and values)` columns are the same as the two previous ones but regarding the secondary suppression. These two indicators depends on the method and are a simple utility metric to compare the methods and are compute as the mean over the 100 simulations. For a same primary suppression pattern, how many additional cells (or values) are suppressed.
- `time comput.` is the mean time in seconds to compute the suppression, including the *merge-and-split* steps for the two concerned methods.

With 10% of primary suppression (table 4), we can notice that, for 10.4% of primary cells representing only 0.8% of the values, the `1-Direct Modular` method is the best in terms of utility, suppressing only 16.8% of cells in average. The second best method is `2-Split Modular` with 19.6% of secondary suppressions in average, followed by `4-Split Hypercube` and far from `3-Direct Hypercube` with more than a third of additional suppressions. Regarding the percentage of values of the secondary suppression, the difference between the three best methods are very small (1 point of percentage) but quite large between these methods and the third one (direct hypercube). In terms of computation time, the two direct methods are almost three times faster. While increasing the percentage of primary suppression in the original tabular data (20% in tab 5 and 50% in tab 6), the observations are quite similar.

As the results displayed here are averages on 100 simulations, we displayed, in figure 3 the distribution of the percentage of values that are suppressed at the secondary step depending on the method and the percentage of primary cells. The `direct hypercube` method is removed from the figure 4 to improve the comparison of the three best options. We can notice that the dispersion of the values is a little bit more important for the hypercube method than for the two `Modular` methods. To complete the analysis, the intervals of values taken by 90% of values over the 100 simulations are displayed in appendix B.

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

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	20.2	2.8	17.0	7.0	4.7
2-Split Modular	2	20.2	2.8	18.3	7.3	10.1
3-Direct Hypercube	1	20.2	2.8	45.5	30.3	3.9
4-Split Hypercube	2	20.2	2.8	20.0	8.1	11.4

Table 5: 4-Dimensional table - split on two non-hierarchical variables - 20% of primary cells

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	50.4	14.2	9.9	8.2	4.2
2-Split Modular	2	50.4	14.2	12.0	9.3	11.4
3-Direct Hypercube	1	50.4	14.2	45.2	78.6	4.3
4-Split Hypercube	2	50.4	14.2	12.4	9.5	11.5

Table 6: 4-Dimensional table - split on two non-hierarchical variables - 50% of primary cells

3.2 Other results

The method to reduce the dimensions of a tabular data suggested in this paper leads to create a potentially high number of tables. The risk of over-suppression increases with the number of sub-tables generated by our method. To test if the risk is real,a simulation on 4-dimensional tables with 2 hierarchical variables used during the *merge-and-split* process was conducted. Both of these variables have 3 nodes, then $2 * 3 * 3 = 18$ sub-tables are created during the *merge-and-split* process. The same

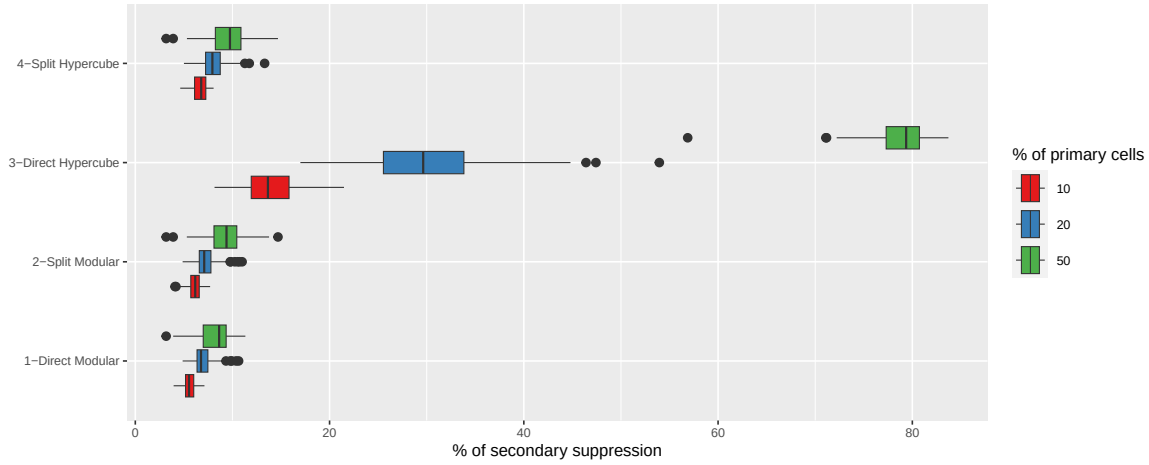


Figure 3: Distribution of % of values suppressed at the secondary step depending on the method and the % of primary cells for a 4-dimensional tabular data with no hierarchical variable.

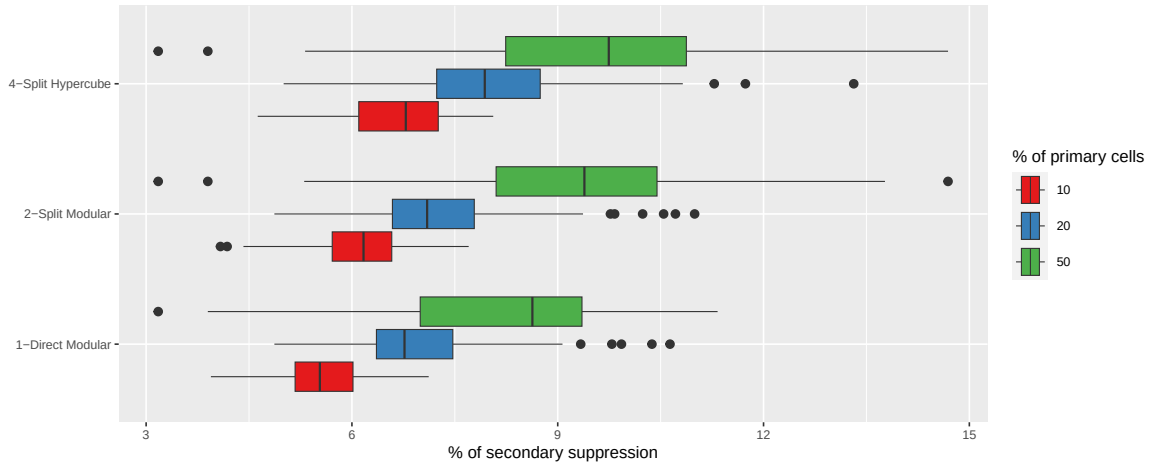


Figure 4: Distribution of % of values suppressed at the secondary step depending on the method (except the Direct `Hypercube` approach) and the % of primary cells for a 4-dimensional tabular data with no hierarchical variable.

simulations as above have been conducted, comparing the same four methods and the same percentages of primary cells. The results for the case with 20% of primary suppression are displayed in the table 7. The conclusions are not so different: even if there are more sub-tables to protect, the over-suppression of our approach is quite limited (5.1 points of percentage in terms of number of cells and 1.5 point in terms of percentage of values).

Regarding 5-dimensional tabular data, the same simulations were conducted. The original tabular data has 5 non hierarchical variables and the `merge-and-split` process creates 4 3-dimensional sub-tables, each having 2 hierarchical variables produced by the process. Since the `direct Modular` approach is impossible with τ−Argus, for such tables only the three others can be compared. The results of the simulations are displayed for the case with 20% of primary suppression in the table 8. The `direct hypercube` method leads to suppress 60.8% of additional cells, which makes the protected table useless. The closeness of the results between the two `split` methods is quite interesting and confirms the results observed on 4-dimensional tables. The figure 5 also shows this closeness in the distribution of the values over the 100 simulations.

All the code to reproduce the results presented in this paper is available on the github repo `https://github.com/InseeFrLab/dims_reduction_tables_workshop_20231215`.

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	20.2	1.2	14.5	2.6	27.6
2-Split Modular	18	20.2	1.2	19.6	4.1	124.4
3-Direct Hypercube	1	20.2	1.2	65.9	72.5	6.2
4-Split Hypercube	18	20.2	1.2	22.2	3.9	128.0

Table 7: Simulations on a 4-Dimensional table - split on two non-hierarchical variables - 20% of primary cells (temporary results on 10 simulations

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

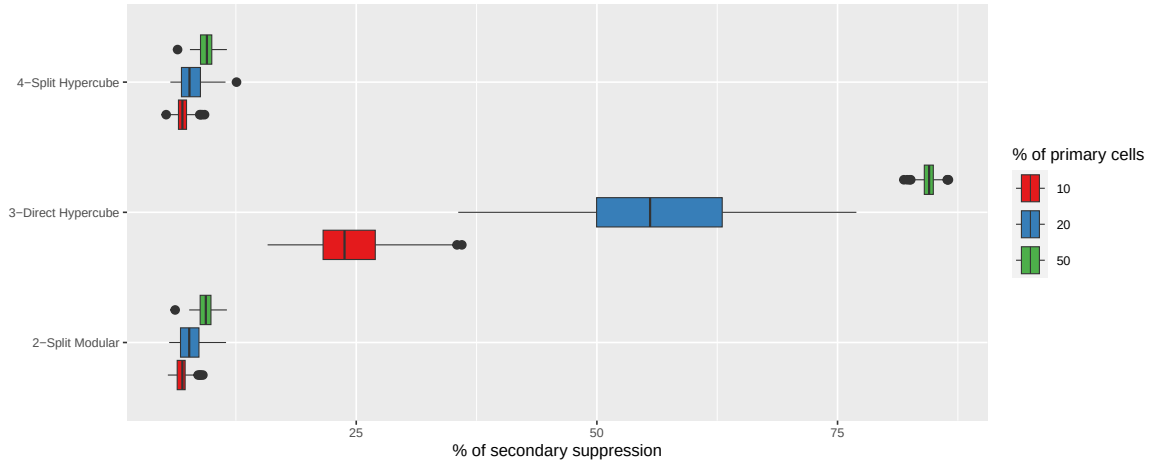


Figure 5: Distribution of % of values suppressed at the secondary step depending on the method and the % of primary cells for a 5-dimensional tabular data with no hierarchical variable.

4 Discussion

The `merge-and-split` method presented in this paper to deal with 4 and 5-dimensional tabular data is theoretically a good way to ensure a protection of such tables with `Modular` implemented in τ-Argus. Indeed, the relevancy of the secondary suppression step of the linked sub-tables produced by the process can be handled by the `tab_multi_manager()` function of the `rtauargus` package. The process creates 3-dimensional tables with new hierarchical variables. The number of linked tables increases with the number of nodes of the merged variables, that's why the implementation in `rtauargus` tends to choose variables to merge so as the least number of tables is produced, when the original table is not too long.

The simulations on 4-dimensional tables shows that the `merge-and-split` method followed by a secondary suppression step with `Modular` is never better than the direct secondary suppression with `Modular`. This was expected. But, the over-suppression generated by the linked tables is quite limited. The same result is noticed with hierarchical variables. Then, we can be confident on the limitation of over-suppression generated by the `merge-and-split` method in the `Modular` case. Nevertheless, the computation time is at least three times greater with this new approach.

In addition, the simulations reveal an unexpected fact: the `merge-and-split` method followed by a secondary suppression step done with `Hypercube` gives results very close from `Modular` and is very different from the direct `Hypercube` approach. If confirmed[3], this could be a very instructive lesson provided by the simulations. Unfortunately, we have to admit that we don't meet this behaviour while testing our tool on more realistic and longer tabular data: the split and the direct approaches with `Hypercube` were, in general, very similar.

In addition of the simulations, we experiment our method on several realistic data and even on several real cases. For "reasonable" tables, the method is very useful and efficient. But when the tabular data has five dimensions tables with several tens of thousands of rows, unfortunately, the `merge-and-split` doesn't make the problems more feasible even if the tables on which `Modular` is applied have only 3-dimensions. And, if feasible, the computation time becomes very great and can be counted in days. Indeed, such large (5 dimensions) and long (tens of thousands of rows) tabular data can contain a lot of zeroes which is one of the main reason why `Modular` isn't able to find a solution in some sub-tables ([De Wolf et al., 2014], p.19). Although long computation times are not a major problem if a single table is released once a year, tabular data is often released in large sets of linked tables. Such computation times become an issue.

[3]The implementation of the simulation has been checked. In particular, the batch files produced by `rtauargus` confirmed that `Hypercube` is applied as expected.

Appendices

A Hierarchies produced while merging X and Y in section 1.2

The `rtauargus::from_4_to_3()` function splits the original table into sub-tables and produces all the hierarchy files corresponding to the `X_Y` variable. This code returns the 30 sub-tables and the 30 hierarchies which are displayed in figures 6 and 7.

```
library(dplyr)
library(rtauargus)
library(sdcHierarchies)

# Construction of the fake data
h <- hier_create(root = "Total", nodes = c("R1","R2"))
h <- hier_add(h, root = "R1", nodes = c("D11", "D12"))
h <- hier_add(h, root = "R2", nodes = c("D21", "D22"))
hier_display(h)

h_graph <- h |> rename(from = root, to = leaf) |> graph_from_data_frame()

act <- hier_create(root = "Total", nodes = c("A","B","C"))
act <- hier_add(act, root = "A", nodes = c("A1", "A2"))
act <- hier_add(act, root = "B", nodes = c("B1", "B2"))
act <- hier_add(act, root = "C", nodes = c("C1", "C2"))
act <- hier_add(act, root = "C1", nodes = c("C21", "C22"))

act_graph <- act |> rename(from = root, to = leaf) |> graph_from_data_frame()

data2 <- expand.grid(
    V1 = c("Total", 1:5), V2 = c("Total", 1:2),
    GEO = unique(c(h$root, h$leaf)),
    ACT = unique(c(act$root, act$leaf)),
    stringsAsFactors = FALSE) |>
  as.data.frame() |>
  mutate(FREQ = sample(1:1000, n(), replace = TRUE))

hier_convert(act, as = "argus") |>
  slice(-1) |>
  mutate(level = substring(paste0(level, name),3)) |>
  select(level) |>
  write.table(file = "act.hrc", quote = FALSE, row.names = FALSE, col.names = FALSE)

hier_convert(h, as = "argus") |>
  slice(-1) |>
  mutate(level = substring(paste0(level, name),3)) |>
  select(level) |>
  write.table(file = "geo.hrc", quote = FALSE, row.names = FALSE, col.names = FALSE)

# Call to the function that merge and split the table
res <- from_4_to_3(
  dfs = data2,
  dfs_name = "data2",
  totcode = c(V1 = "Total", V2 = "Total", GEO = "Total", ACT = "Total"),
  hrcfiles = c(GEO = "geo.hrc", ACT = "act.hrc"),
  v1 = "GEO",
  v2 = "ACT"
)
```

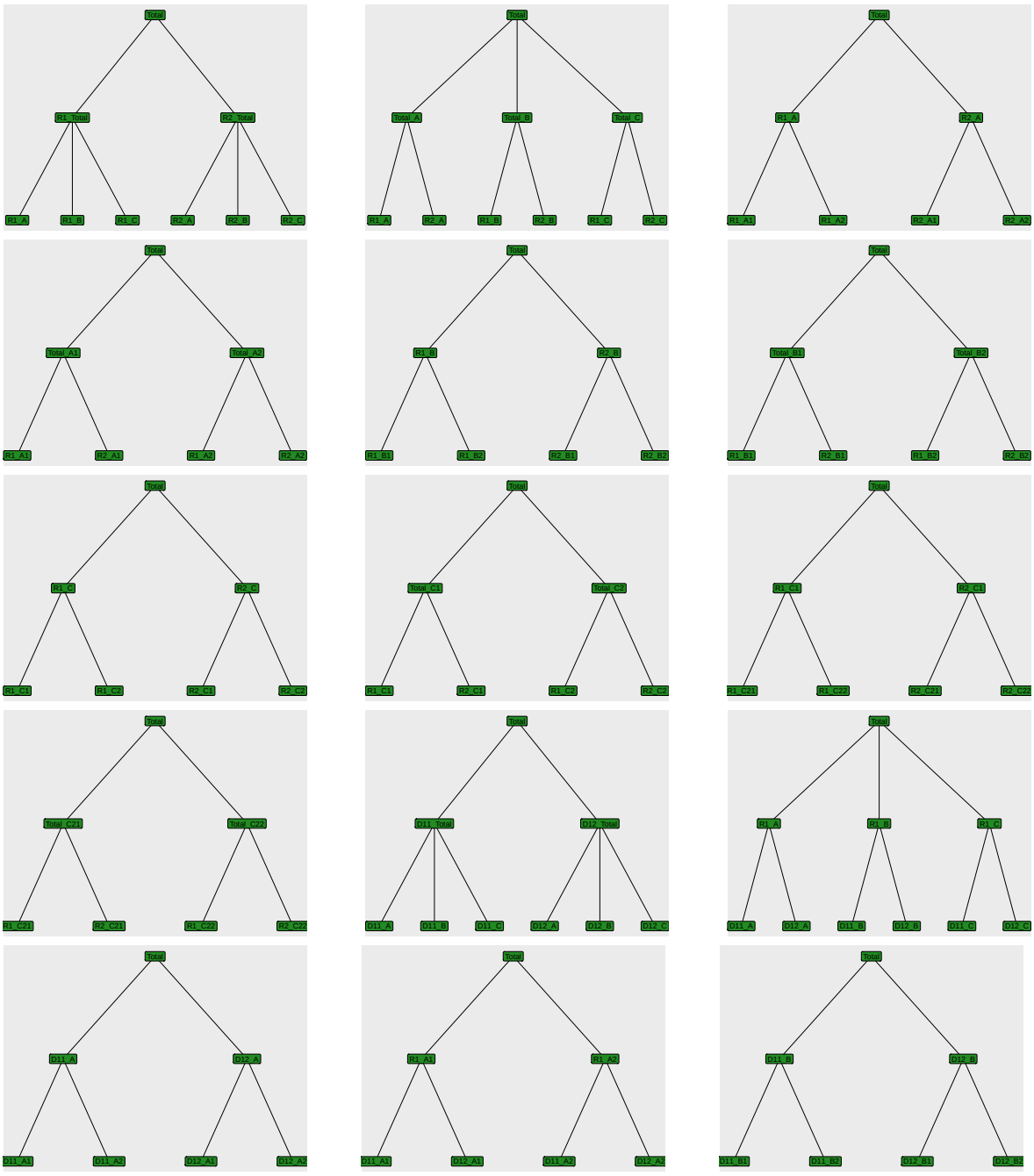


Figure 6: Hierarchies of the merged variable X_Y for each sub-table (1)

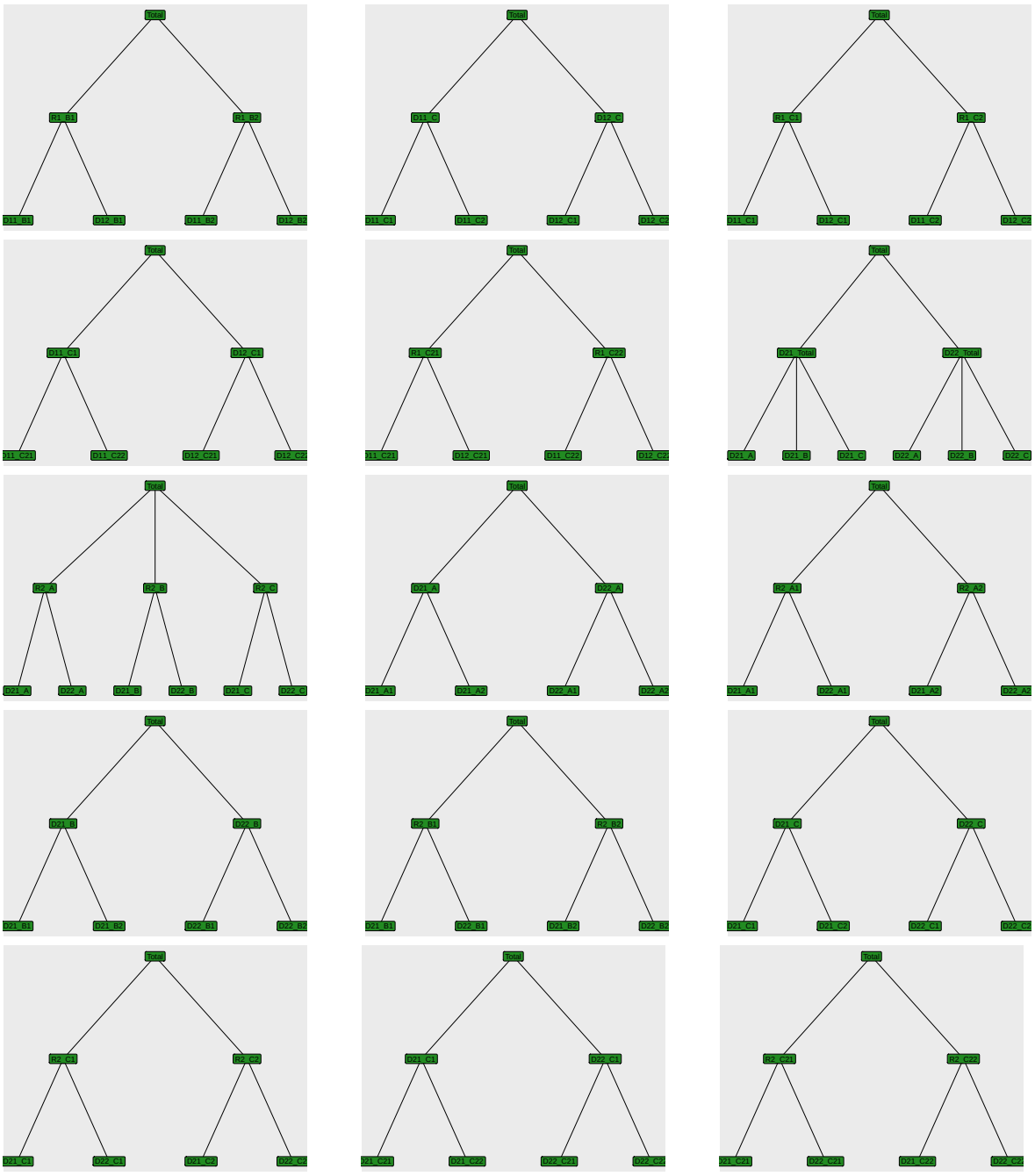


Figure 7: Hierarchies of the merged variable X_Y for each sub-table (2)

B Monte-Carlo Confidence Intervals at 90% level of confidence

	primary suppression		secondary suppression		
method	n cells (%)	values (%)	n cells (%)	values (%)	time computation
1-Direct Modular	[10.2; 11.1]	[0.7; 1.1]	[14.6; 19.6]	[4.6; 6.4]	[3.1; 5.6]
2-Split Modular	[10.2; 11.1]	[0.7; 1.1]	[16; 22.2]	[5; 7.3]	[9.1; 14]
3-Direct Hypercube	[10.2; 11.1]	[0.7; 1.1]	[28; 40.4]	[10; 18.3]	[3; 4.9]
4-Split Hypercube	[10.2; 11.1]	[0.7; 1.1]	[16.9; 24.5]	[5.2; 7.8]	[9.5; 15.1]

Table 9: 4-Dimensional table - split on two non-hierarchical variables - 10% of primary cells

	primary suppression		secondary suppression		
method	n cells (%)	values (%)	n cells (%)	values (%)	time computation
1-Direct Modular	[20; 20.9]	[2.3; 3.3]	[14.2; 20]	[5.7; 9.1]	[3.3; 8.5]
2-Split Modular	[20; 20.9]	[2.3; 3.3]	[15.5; 21.4]	[6; 9.8]	[8.9; 12.6]
3-Direct Hypercube	[20; 20.9]	[2.3; 3.3]	[38.6; 52.5]	[20.1; 44]	[3.1; 5.3]
4-Split Hypercube	[20; 20.9]	[2.3; 3.3]	[16.4; 23.6]	[6.4; 10.2]	[9.2; 15.5]

Table 10: 4-Dimensional table - split on two non-hierarchical variables - 20% of primary cells

	primary suppression		secondary suppression		
method	n cells (%)	values (%)	n cells (%)	values (%)	time computation
1-Direct Modular	[50.2; 51.1]	[13.7; 14.7]	[6.7; 12]	[5.7; 10]	[3.2; 5.6]
2-Split Modular	[50.2; 51.1]	[13.7; 14.7]	[8.4; 15.6]	[6.2; 12.5]	[9.7; 15.4]
3-Direct Hypercube	[50.2; 51.1]	[13.7; 14.7]	[42.7; 47.6]	[72.7; 82.5]	[3.5; 5.6]
4-Split Hypercube	[50.2; 51.1]	[13.7; 14.7]	[8.4; 15.6]	[6.3; 12.3]	[9.9; 15.1]

Table 11: 4-Dimensional table - split on two non-hierarchical variables - 50% of primary cells

References

[Berrard et al., 2023] 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, 2002] 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 et al., 2014] 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, 1994] Repsilber, R. D. (1994). Preservation of confidentiality in aggregated data. In *Second International Seminar on Statistical Confidentiality*.