Przegląd Statystyczny. Statistical Review, 2021, vol. 68, 4, 1–20 DOI: 10.5604/01.3001.0015.7797
Received: 20.12.2021, revised: 9.02.2022, accepted: 15.02.2022

Comparison of methods used for filling partially unobserved contingency tables

Michał Kot,[a] Bogumił Kamiński[b]

Abstract. In this article, we investigate contingency tables where the entries containing small counts are unknown for data privacy reasons. We propose and test two competitive methods for estimating the unknown entries: our modification of the Iterative Proportional Fitting Procedure (IPFP), and one of the Monte Carlo Markov Chain methods called Shake-and-Bake. We use simulation experiments to test these methods in terms of time complexity and the accuracy of searching the space of feasible solutions. To simplify the estimation procedure, we propose to pre-process partially unknown contingency tables with simple heuristics and dimensionality-reduction techniques to find and fill all trivial entries. Our results demonstrate that if the number of missing cells is not very large, the pre-processing is often enough to find fillings for the unknown values in contingency tables. In the cases where simple heuristics are insufficient, the Shake-and-Bake technique outperforms the modified IPFP in terms of time complexity and the accuracy of searching the space of feasible solutions.

Keywords: contingency tables, Markov Chain Monte Carlo, Iterative Proportional Fitting Procedure
JEL: C15, C44

1. Introduction

The study of dependencies between variables is one of the key aspects of data analysis. The procedure of verifying associations between variables depends on the type of variables. In the case of ordinal or nominal variables, it is common to investigate this relationship by their joint frequency distribution provided by a multidimensional contingency table. The elements of such a table (referred to as entries or cells) contain the frequency of joint occurrences of the outcomes of all the underlying variables. Contingency tables, which are the focus of this article, are considered to be a simple and effective technique to analyse data (Payne & Payne, 2011), and are utilised in e.g. social sciences (Payne & Payne, 2011), medical research (Zelterman & Louis, 2019) or biology (Bailey, 1995).

In practical applications, due to confidentiality reasons, data aggregated in the form of a contingency table can be incomplete, and thus not useful. This problem occurs because certain combinations of features included in a contingency table may

[a] SGH Warsaw School of Economics, Institute of Econometrics, Decision Analysis and Support Unit, ul. Madalińskiego 6/8, 02-513 Warszawa, e-mail: mk46032@sgh.waw.pl, ORCID: https://orcid.org/0000-0002-7885-4028.
[b] SGH Warsaw School of Economics, Institute of Econometrics, Decision Analysis and Support Unit, ul. Madalińskiego 6/8, 02-513 Warszawa, e-mail: bkamins@sgh.waw.pl, ORCID: https://orcid.org/0000-0002-0678-282X.

be so rare that they can potentially disclose personal or sensitive data (Slavković, 2010; Slavković & Lee, 2010). In such a case, the end user of the data presented in the form of a contingency table is provided with information only on the marginal distributions of features along with cell sizes big enough to ensure data privacy. The problem with audiences being too small to be reported, common in marketing, is a good example of the above. For instance, Nielsen does not report TV ratings for audiences of sizes below the assumed level (Eastman & Ferguson, 2012), and neither Facebook nor Google show advertisements to custom audiences unless their size requirement is met (Facebook.com, n.d.; Google.com, n.d.). Such practice results from the fact that data providers striving to comply with the General Data Protection Regulation (GDPR) face a trade-off between securing data privacy and ensuring data utility (Slavković, 2010).

We encountered the same problem during the construction of an agent-based model for the calculation of the cross-media reach of advertising (Kot & Kamiński, 2021). When designing the model, we assumed that the synthetic population of agents reflects the existing population of Poland in terms of the agents' socio-demographic features. Since certain dimensions are correlated with others, sampling each feature independently would lead to a bias in the model. Therefore, the necessary condition for the agents' sampling procedure was to obtain a contingency table of features where the probabilities of all combinations would be known. Since we lacked a complete contingency table, we began to seek a solution to this problem.

The literature points to the Iterative Proportional Fitting Procedure (IPFP) as the solution to the problem of estimating unknown entries of contingency tables based on known marginal distributions (Bacharach, 1965; Deming & Stephan, 1940). The IPFP allows the estimation of the sizes of the empty cells that match marginal restrictions by the iterative adjustment of the sizes of missing entries across all the dimensions of the problem, until the marginal totals converge to the target ones. The starting point for the estimation is the construction of a matrix of the initial sizes of all entries, which reflect the prior information available to the researcher. A detailed discussion on the mechanics of the IPFP can be found in Lovelace et al. (2015).

Recent state-of-the-art methods of estimating contingency tables are presented in the literature on issues related to data disclosure. Research in this field was initiated by Diaconis and Sturmfels (1998). In their article, the authors proposed generating contingency tables using the Markov moves. They start with the initial contingency table meeting the known marginal requirements. Then, the Markov moves applied to this table introduce pairwise integer changes to the cells' sizes, resulting in the emergence of a new contingency table, where the marginal requirements remain fulfilled. The new table is accepted if all its entries are non-negative. A finite collection of Markov moves linking contingency tables with the same marginals is called a Markov basis (Dobra, 2003). However, the Markov basis in an explicit form is usually applied to only a few problems because of high computational costs related to its generation (Aoki et al., 2012). Instead of generating the entire Markov basis, Dobra (2012) proposes constructing a Markov chain of locally connected contingency tables. The resulting Markov chain is a subset of all the feasible tables. In this approach, it is possible to introduce additional constraints regarding the lower and upper boundaries of the cells.

Despite the indisputable advantages of the above-mentioned methods, we want to emphasize that three aspects of our problem make it different from the issues usually solved by them.

Firstly, the primary motivation of our research was to use completed contingency tables in the process of sampling the population of artificial agents. To meet this end, we were looking for a method to efficiently generate tables that were uniformly distributed on a set of feasible contingency tables. Sampling a population of agents from the set of contingency tables which differed from each other in terms of the estimated values of the unknown cells improved the results of the simulation study. It is because the conclusions drawn from the simulation results are valid for any feasible combination of features in the population, not only for a series of local, similar contingency tables.

The second unique feature of our problem was that only some elements of the contingency table were unknown. Other elements with a sufficiently high count were observed and had to remain fixed.

The final aspect of our problem was that we had no prior knowledge regarding the missing cells' entry size, except that they were non-negative and of a smaller or equal size to the known value (referred to as threshold). Thus, in our case, the set of feasible solutions was limited by the knowledge of the following features of the problem under study: (1) the marginal distributions of variables captured by the contingency table, (2) the sizes of entries with a sufficiently high count, and (3) the minimal required size of cells to become observable. In the next two paragraphs we will discuss how the unique structure of our problem affected the utility of the discussed algorithms.

In the case of the classic version of the IPFP, all entries in the contingency table are subject to change in each iteration of the algorithm. Therefore, no fixed cells are allowed, except for entries of the value of 0. The algorithm of Diaconis and Sturmfels, which could potentially be used in our case, has two shortcomings. Firstly, all the steps of the procedure are complicated and time-consuming, as they require solving several instances of optimisation problems. Secondly, as the method uses local Markov moves, it does not provide information whether the whole Markov basis was visited after a given number of steps.

At the same time, we are aware that our specific problem is not the only area that these methods may be applied to, i.e. they can be implemented to efficiently solve a wider spectrum of problems.

Since the use of the classic IPFP or the Markov moves would not allow the achievement of the goals listed in the previous paragraph, we decided to test other solutions which allowed the coverage of the searched area of the solution space and the time complexity. In the process of examining the competitive methods, we will use a simulation approach described in Section 3. Firstly, we will generate contingency tables with unobserved cells. Secondly, we will pre-process the contingency table by searching for all trivial cells' sizes (which in a few cases will allow the whole contingency table to be solved). Finally, if a contingency table still contains unobserved cells, we will try to address the situation with the methods listed in the next two paragraphs.

The first method we will test involves our modification of the IPFP algorithm, which has the capacity to find feasible solutions in a contingency table with fixed entries (a description of this modification is presented in detail in Section 2.3).

The second tested method is the Markov Chain Monte Carlo technique known as Shake-and-Bake (SB) (Boender et al., 1991), capable of approximating the distribution of the solution set by an asymptotic sampling of uniform points on a boundary of a convex polytope.

Despite being similar in terms of generating floating-point solutions, there are significant differences between the two competitive methods. In the modified IPFP, we are able to control the matrix of the initial weights only, which enables us to force specific entries' sizes to 0, but we cannot ensure that all the entries will meet the upper boundary requirement. On the other hand, in the SB, we do not assume the existence of any fixed relation between variables. Moreover, to reveal the size of a cell, we can apply prior knowledge regarding the minimum and maximum feasible value (or more generally, some more complex prior knowledge).

In order to test the coverage of the searched area of the solution space, we need a reference solution with all possible fillings of a given contingency table. For objective reasons, we can provide such a solution only for smaller-scale problems through the introduction of a pre-processing stage based on dimensionality reduction methods. We propose to consider exhaustive enumeration of the missing entries of a contingency table as the solution to a set of linear equations. To find all the possible non-negative integer solutions, we will use constraint programming. It is worth emphasizing that exhaustive enumeration and sampling are, along with the computation of sharp integer bounds and counting, the directions in which the research on contingency tables is currently heading (Dobra & Fienberg, 2010).

Our results show how to effectively sample solutions for partially unknown multidimensional contingency tables. This procedure is efficient mostly thanks to two elements of our analysis: the array pre-processing, based on heuristics and inspirations from the hypergraph theory, which results in a significant reduction of the problem's size and time complexity, and a detailed comparison of our modification of the IPFP and SB algorithms in terms of their ability to cover the space of feasible solutions and time required to return a given volume of samples.

The further part of the article consists of: Section 2, where we formulate the problem and describe the details of the tested algorithms, Section 3, where the planning of the simulation experiments and the results of the simulations' run in relation to benchmark methods are shown, and Section 4, in which we discuss the potential limitations of the used methods and propose the directions for their future development.

2. Formulation of the problem and the methods used

In the first part of this section, we formulate the research problem in detail. In the second part, we discuss the pre-processing of the problem. In the third part, we present the method of searching for all the solutions which served as a reference point for the two competitive methods applied in small-scale problems. Finally, we describe the methods that can be used to solve the research problem: in Section 2.4, we present our modification of the standard IPFP approach, and in Section 2.5 the SB algorithm.

2.1. Formulation of the research problem

We assume that a population of an N size is split by K features, with k_i states each. The split generates a Cartesian product (contingency table) with $\prod_{i=1}^{K} k_i$ entries. For simplicity, we assume that feature i takes values from set $V_i = \{1,2,\dots,k_i\}$. By p_m we denote the size of cell $m \in V_1 \times\ldots\times V_K$. We assume that all p_m are non-negative. We also assume that we know all the marginal totals, which for dimension i are denoted by $p_v{}^i$ for $v \in V_i$. We assume that cells are observable if and only if their size is greater than threshold level T that is known to the researcher. Therefore, in terms of dimensions, the table of observable cells is identical to the complete table, with entries q_m structured as in the following equation:

$$q_m = \begin{cases} p_m \; if \; p_m > T \\ 0, otherwise \end{cases}. \tag{1}$$

Similarly, we denote the marginal distributions of the known cells for value $v \in V_i$ in dimension i as $q_v{}^i$, and by subtracting them from $p_v{}^i$, we obtain marginal boundaries for unobserved cell sizes ${r_v}^i = {p_v}^i - {q_v}^i$. An example of the calculation of ${r_v}^i$, ${p_v}^i$ and ${q_v}^i$ can be found in Table 1. The goal of the algorithm we want to develop is to find all values of $\hat{p}_m$ that solve the system of linear equations:

$$\begin{gathered} \sum_{m \in {R_V}^i} \hat{p}_m = {r_v}^i \\ \hat{p}_m \geq 0 \\ \hat{p}_m \leq T \\ \hat{p}_m \in \mathbb{Z} \end{gathered} \quad . \tag{2}$$

In Equation (2), ${R_V}^i$ is a set of indices in $m \in V_1 \times \ldots \times V_K$ such that $\forall m \in {R_V}^i : q_m = 0$ and in the i-th dimension they are of v value. Based on the example shown in Table 1, the unknown elements are $\hat{p}_{2,2}$, $\hat{p}_{2,3}$, $\hat{p}_{3,2}$ and $\hat{p}_{3,3}$.

Table 1. Example of the ${p_v}^i$, ${q_v}^i$ and ${r_v}^i$ calculation for $T = 15$

				${p_v}^1$	${q_v}^1$	${r_v}^1$
	30	20	20	70	70	0
	25	$\hat{p}_{2,2}$	$\hat{p}_{2,3}$	40	25	15
	20	$\hat{p}_{3,2}$	$\hat{p}_{3,3}$	35	20	15
${p_v}^2$	75	30	40			
${q_v}^2$	75	20	20			
${r_v}^2$	0	10	20			

Source: authors' calculations.

Note. values $p_{2,2}$, $p_{2,3}$, $p_{3,2}$, and $p_{3,3}$ are not observed.

The marginal restrictions for this example are presented in Equation (3):

$$\begin{gathered} \hat{p}_{2,2} + \hat{p}_{2,3} = 15 \\ \hat{p}_{3,2} + \hat{p}_{3,3} = 15 \\ \hat{p}_{2,2} + \hat{p}_{3,2} = 10 \\ \hat{p}_{2,3} + \hat{p}_{3,3} = 20 \end{gathered} \quad . \tag{3}$$

2.2. Pre-processing of the problem

To reduce the search space of the calculations, we introduce a pre-processing stage to be run before attempting to identify $\hat{p}_m$. Its main purpose is to find all trivial solutions to a given problem and therefore to accelerate the calculations. During the pre-processing stage, we iteratively search through the table of observable cells for:

- single missing cells in some dimensions (i.e. the cases where $|R_V{}^i| = 1$), where we input r_v in these dimensions;
- dimensions in which unknown entries exist and the marginal sum is 0, where we input 0 in each missing entry;
- dimensions in which the number of missing cells multiplied by the threshold value is equal to the marginal sum of the unknown cells (i.e. $|R_V{}^i| \cdot T = r_v{}^i$), where we input the threshold value in each missing entry.

Moreover, we extend the pre-processing stage described above to further simplify the calculations. Inspired by the hypergraph theory (Bretto, 2013), we treat the set of the remaining missing cells $\{m\colon q_m = 0\}$ as vertices of a hypergraph. The hyperedges of the hypergraph are defined by sets $R_V{}^i$. We seek a set of vertices V that separates the underlying total hypergraph into independent subhypergraphs. We assume set V to be minimal, i.e. the removal of any element of V results in V no longer separating the hypergraph. Then, the potential values of the cells in the separated subhypergraphs are conditioned on the values of the separating vertices in V. This approach significantly limits the solution space. Below we provide examples of how this procedure simplifies the problem and accelerates the computation process.

The first example of such a situation is presented in Table 2, where the threshold level is equal to $T = 20$. Without the separation, it can be computed that an algorithm should find feasible solutions among 441 potential combinations. One can notice that the p_{32} vertex can be used to separate the problem into two distinct problems that are easier to solve. Furthermore, only $p_{32} = 10$ allows all marginal constraints to be met and thus the size of the feasible set is reduced to 42 combinations.

Table 2. A contingency matrix where p_{32} is a separating vertex

p_{11}	p_{12}	30	30	80
p_{21}	p_{22}	30	30	80
30	p_{32}	p_{33}	p_{34}	60
30	30	p_{43}	p_{44}	80
80	60	80	80	300

Source: authors' calculations.

The second example is presented in Table 3, where the threshold level is equal to $T = 30$. It can be computed that without the separation, it would require finding feasible solutions among 1,224,912 potential combinations. The assumption that $p_{4,3} = 10$ (which, as above, can be proven to be the only admissible entry) results in an 80%-reduction of a combination set, to 122,491×2 elements, which in practice is much easier to compute.

Please note that in certain situations the pre-processing stage can solve the array and fill all missing entries; if it is not able to solve the array, the missing cells form a subarray of at least 2 elements across each dimension (the smallest possible is a 2×2 subarray).

Table 3. A contingency matrix where p_{43} is the separating vertex

p_{11}	p_{12}	p_{13}	40	40	40	150
p_{21}	p_{22}	p_{23}	40	40	40	150
p_{31}	p_{32}	p_{33}	40	40	40	150
40	40	p_{43}	p_{44}	p_{45}	p_{46}	120
40	40	40	p_{54}	p_{55}	p_{56}	150
40	40	40	p_{64}	p_{65}	p_{66}	150
150	150	120	150	150	150	870

Source: authors' calculations.

2.3. A method searching for all solutions

In this subsection, we discuss the algorithm used as a reference for the tested methods, which relies on finding all feasible integer solutions to a problem expressed by Equation (2). Since the feasible set of solutions is in this case limited by marginal sums across all dimensions, one can think of solving the contingency table as of solving a set of linear equations. In practice, to solve the linear system, we use software that can be used to solve constraint programming problems called MiniZinc (Nethercote et al., 2007).

Since the exhaustive enumeration of all solutions is highly time-consuming even for specialised software, we will use it as a reference for small-scale problems only.

An example presented in Table 4 is used to show the impact of different threshold levels on the estimation process. As we have previously assumed, the threshold level is known, i.e. the researcher is aware that all the sizes of unknown entries are smaller than some fixed value T. If the threshold was unknown, the presented problem would have 9 feasible solutions presented in Table 5. If the threshold level was set at 9, only 7 results would be valid, because answers #8 and #9 would contain elements greater than the threshold value. By the same token, for threshold levels 10 and 11, there would be 8 and 9 solutions, respectively.

Table 4. A contingency matrix without known cells

p_{11}	p_{12}	8
p_{21}	p_{22}	12
9	11	20

Source: authors' calculations.

Table 5. Solutions to the problem presented in Table 4

ID	p_{11}	p_{12}	p_{21}	p_{22}
#1	0	8	9	3
#2	1	7	8	4
#3	2	6	7	5
#4	3	5	6	6
#5	4	4	5	7
#6	5	3	4	8
#7	6	2	3	9
#8	7	1	2	10
#9	8	0	1	11

Source: authors' calculations.

2.4. Modified IPFP method

To present how we have modified the classic IPFP, we should start with discussing the mechanics of the classic version of this algorithm based on an example for the 2×2 contingency table presented in Table 6. Consider an unknown contingency table and assume that the target row and the column totals are (25, 35) and (20, 40), respectively. We have no prior knowledge about the entry sizes, thus we initially assume them to be equal to 1. The first step of each IPFP iteration creates multipliers to adjust the sum of the row elements in an entry table to the marginal totals. Then, the IPFP repeats the procedure with the column totals. In an extended case of a multidimensional array, multipliers are created along each dimension and the procedure follows the two-dimensional example. In consecutive iterations, the same procedure is repeated until the algorithm converges, i.e. the values in the cells change from iteration to iteration by less than the assumed tolerance level. The convergence of the IPFP algorithm was proven by Fienberg (1970) in the case of strictly positive tables. As regards non-negative tables where certain cells equal 0, convergence was proven by Csiszár (1975). In the presented example, the IPFP converges after 3 iterations, and the initial and final steps are presented in Tables 6 and 7, respectively.

As mentioned before, the problem considered in this paper is different from those normally solved by the classic IPFP. Therefore, we had to modify its normal specification to be able to use this method. Initially, we know all the fixed cells' sizes and the marginal totals. In the first step, we subtract the sums of the known cells across each dimension from the known marginal totals in order to obtain the sums for the unknown cells. In the second step, we create an array of the initial values – if the cell is known, then the initial value will be positive or otherwise equal to 0, resulting in an array with non-negative values. Our objective is to sample the possible fillings. A single run of a modified IPFP algorithm with the same initial weights would produce similar results. Thus, in our case, we propose applying a multi-start of the algorithm with random initial weights from the standard uniform distribution.

The adjustment of the cell values across all dimensions is performed in a similar way to the standard version of the IPFP algorithm, yet the target values are the marginal sizes of the unknown entries. The algorithm is completed when the convergence requirement is met, i.e. the highest multiplier across all dimensions does not exceed the assumed value. Since our initial array is non-negative and contains 0, the proof of the convergence of the modified IPFP algorithm follows from Csiszár (1975).

Table 6. Example of classic IPFP mechanics – initial table

1	1	25
1	1	35
20	40	60

Source: authors' calculations.

Table 7. Example of classic IPFP mechanics – final table

$8\frac{1}{3}$	$16\frac{2}{3}$	25
$11\frac{2}{3}$	$23\frac{1}{3}$	35
20	40	60

Source: authors' calculations.

2.5. Shake-and-Bake simulated result

The second competitive method that we use for solving problems with a high number of missing values is Shake-and-Bake, which is based on the Monte Carlo Markov Chain technique (Boender et al., 1991).

SB generates uniform points from the boundary of convex polytope L, limited by a set of m linear inequalities $\boldsymbol{Ax} \leq \boldsymbol{b}$. The method also assumes lack of redundant equations, and the boundary of polytope ∂L is defined as a set of points, for which only one inequality constraint is active at a time as per Equation (4):

$$\partial L = \bigcup_{i=1}^{m} \left\{ \boldsymbol{x} : \boldsymbol{a}_i{}^T \boldsymbol{x} = b_i, \boldsymbol{a}_j{}^T \boldsymbol{x} = b_j, \forall j \neq i \right\}. \tag{4}$$

The mechanics of the SB algorithm can be presented in a few steps. The beginning of the algorithm is an arbitrary point on the boundary of the polytope. Then, random feasible search direction vector d is sampled, and the algorithm jumps from the starting point in a direction defined by d to a target point. The target point is defined as the closest to the starting point intersection of the line defined by the starting point, search direction vector and one of the polytope boundaries. All iterations undergo the described process. The detailed presentation and discussion of the SB algorithm can be found in Kroese et al. (2011).

Note that the problem described by Equation (2) meets the conditions of the SB algorithm if we remove the constraint due to the fact that the solutions must be an integer.

In this section, we defined the research problem and described the pre-processing of an incomplete contingency table that helps to simplify the research problem. We introduced three methods that will be used in simulations: (1) the method which enumerates all feasible integer fillings used as a reference point, (2) our modification of the IPFP, and (3) the SB method. Despite the fact that two of the tested methods sample feasible float fillings, their mechanics differ significantly. The modified IPFP applies a multi-start approach, which in each iteration starts from different uniform weights and follows a procedure of iterative adjustment across all of the array's marginals (as described in Section 2.4) to return a feasible solution. The SB approach, on the other hand, creates a convex polytope that represents the problem's constraints and iteratively jumps on its boundaries to produce a sample of feasible fillings. What is more, the modified IPFP allows the setting of only the lower boundary for solutions, while in SB both the lower and upper boundaries can be set.

3. Simulation setup and results

In this section, we use simulations to test the ability of the modified IPFP, and the SB methods to cover the searched area of the solution space and time complexity. To compare the coverage of the feasible area, we need a reference solution – the result of the method described in Section 2.3 listing all the possible integer fillings. Since the number of solutions grows rapidly with the increasing size of the contingency table, we run this test on two-dimensional arrays only, with a limited number of missing cells. In the first part of this section, we describe the simulation procedure, and in the second we present the results.

3.1. Simulation procedure

The method used to compare the performance of the competitive techniques firstly involved the sampling of the multidimensional contingency table. We assumed that the table would contain information about an artificial population of N units. Each unit was described by K nominal variables, with k_i states in each variable. Firstly, we sampled a random realisation of a standard uniform distribution for each cell of the table, which served as the weight of a given combination of features. Then, each unit of the population was randomly assigned to a given cell proportionally to the weights, sampled in Step 1. Therefore, each unit of the population was put into a single cell of the contingency table. Finally, we chose threshold T as percentile c of the joint distribution stored in the contingency table, and all cells of sizes smaller or equal to T were marked as missing. As a result of the above procedure, we received the known marginal distributions of a full array, the incomplete array of observable cells, and the threshold.

We planned the simulation procedure in such a way as to capture the coverage of the search area and the time required by each method to generate 1,000 solutions according to various parameters. We tested different parameters, including: (1) the marginal sizes k_i of the contingency tables (the number of the distinct states of a nominal variable), (2) the number of dimensions (features) K, (3) the number of units in population N, and (4) the threshold level in the form of a percentile of joint distribution c. Having managed not to lose generality, we sampled only hypercube contingency tables with equal sizes on each margin ($k_i = k$). We split the simulation procedure into three independent experiments, each focusing on different aspects of the investigated problem.

The aim of the first experiment was to analyse the pre-processing procedure. The purpose of the second experiment was to investigate the time complexity of the two competitive methods. Finally, in the third experiment, we compared the coverage of the searched area of the feasible solutions. Table 8 presents the ranges of the parameters which were set for each experiment.

Table 8. Ranges of simulation parameters

Parameter	Experiment 1	Experiment 2	Experiment 3
k_i	4, 5, 6, 7, 8, 9	6, 7, 8, 9	8
K	2, 3, 4	2, 3	2
N	10* k_i *K	10* k_i *K	640
c	0.10, 0.15, 0.20, 0.25, 0.30	0.25, 0.30	0.20
Number of simulations	11,000	2,000	1,000

Source: authors' calculations.

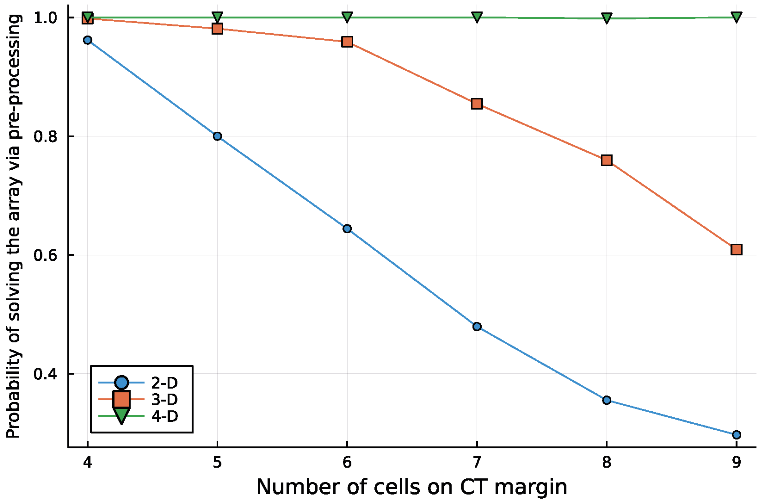
A single iteration run consisted, in the case of Experiment 1, of three steps, while for Experiments 2 and 3 it involved four steps. In the first step, we sampled k_i, K and c. In the second step, we sampled a contingency table for parameters k_i and K obtained in Step 1, and trimmed the entries below the threshold level, defined by c. In the third step, we pre-processed the incomplete contingency table to input all the trivial fillings. In the fourth step, we used the modified IPFP (with initial weights being randomly uniform) and SB algorithms to sample 1,000 possible fillings of the remaining unknown entries. In Experiment 3, we also searched for all the possible integer fillings to investigate the coverage of the searching of two sampling-based methods. In each experiment, we ran a simulation several times (please refer to row *Number of simulations* in Table 8) to obtain stable results. It is worth remembering that the modified IPFP and SB sample floating-point numbers. To compare their results with the integer solutions, we rounded them and ensured that the rounding does not bring solutions outside of the feasible set.

The simulations have been conducted in Julia language (Bezanson et al., 2017), where we developed all the methods used in this paper: the modified IPFP, the exact method incorporating MiniZinc (Nethercote et al., 2007) and SB using R language (R Core Team, 2018), and the hitandrun package (van Valkenhoef & Tervonen, 2019).

3.2. Simulation results

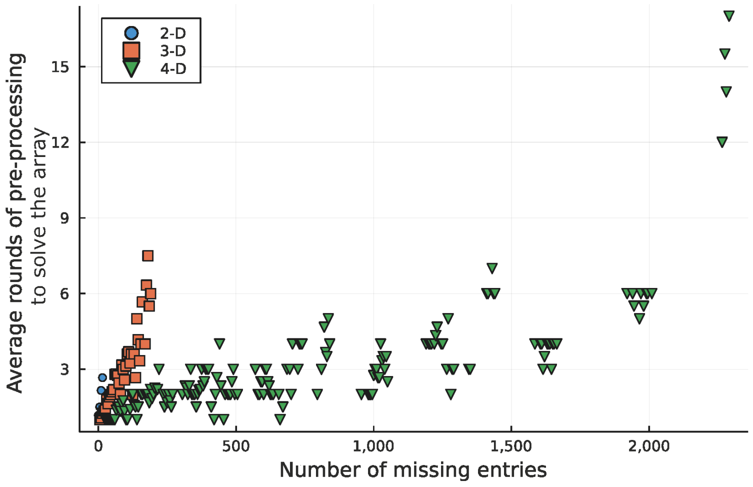
The results obtained in the first experiment showed a relation between the problem's size (defined by the number of array dimensions) and the probability that the pre-processing managed to solve it. We observed that a higher number of dimensions increases the probability that certain unknown entries could be revealed with simple heuristics of a pre-processing stage. For example, if the marginal size is equal to 7, the probability that pre-processing will solve a two-dimensional table is 0.48, while for three- and four-dimensional tables it is 0.85 and 1.00, respectively. At the same time, more complex problems with a higher number of unknown entries require a higher number of pre-processing rounds to be solved. On average, two-dimensional problems require 1.58 rounds to be solved, while three- and four-dimensional ones require 2.35 and 2.58 rounds, respectively. Figures 1 and 2 present the results of Experiment 1, which confirm the two above observations.

Figure 1. Probability of solving the array by pre-processing



Source: authors' calculations.

Figure 2. Number of pre-processing rounds necessary to solve the array

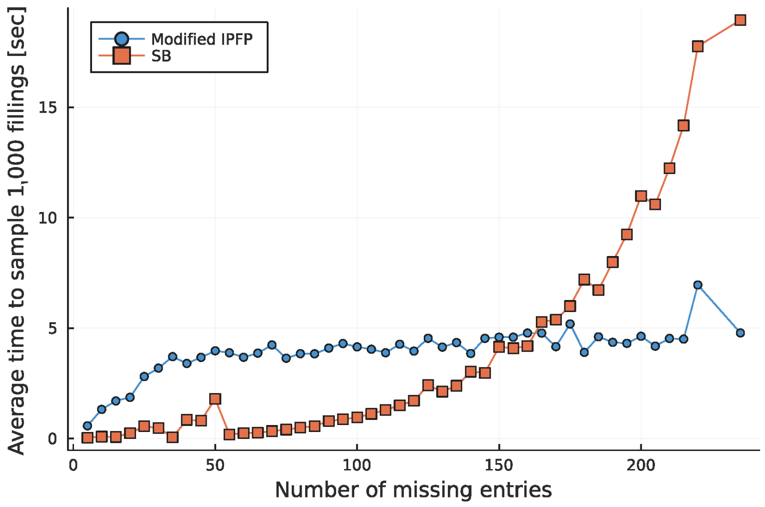


Source: authors' calculations.

In the second experiment, we tested two competitive methods, i.e. our modified version of the IPFP and SB, in terms of time complexity. The results, presented in Figure 3, show that SB requires less time to generate 1,000 samples than the modified

IPFP needs to solve problems of a small or moderate size (up to 150 unknown entries). As the problem's size increases, the amount of time required for the modified IPFP to solve the problem increases, but only up to 50 missing entries, and it remains on a similar level for more complex problems. In the case of SB, time complexity grows exponentially along the problem's complexity. Since complex problems with a high number of missing counts occur less often due to the pre-processing stage, the average time for the modified IPFP to sample 1,000 solutions was 2.76 seconds, while for SB it was 1.01 seconds.

Figure 3. Time complexity of the modified IPFP and SB



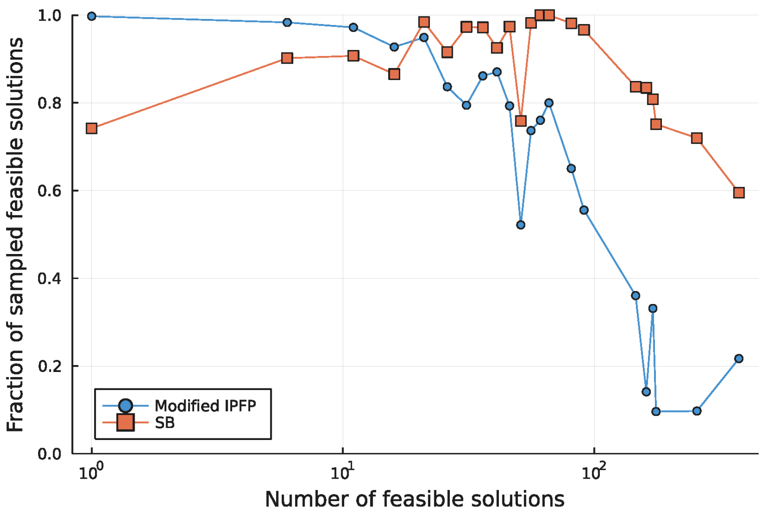
Source: authors' calculations.

The third experiment was designed to compare the ability of both methods to effectively search through the space of feasible solutions. An optimal method should be able to meet two requirements: to sample each possible filling and to generate a uniform sample where all fillings occur equally often.

We present the results of an experiment relating to the first requirement in Figure 4. The modified IPFP can sample all feasible solutions for problems to which there are fewer than 50 solutions. SB outperforms the modified IPFP, as it can sample all solutions for problems with 100 feasible solutions. In the case of problems with over 100 feasible solutions, however, none of the above methods is able to effectively sample all possible fillings.

As regards the second requirement, we propose testing the sampling uniformity with two metrics, the first of which is the ratio of the number of occurrences of the solutions sampled most often to the number of occurrences of solutions sampled least often, as displayed in Figure 5. For example, if the ratio is equal to 10, a given algorithm samples the most frequent solution ten times as often as the least frequent one. In the case of the modified IPFP and SB, the first uniformity metric increases as the number of feasible solutions grows up to 30 and 50, respectively, and then sharply decreases. The shapes of both curves result from the fact that for problems with a low number of feasible fillings, it is more probable that both will be sampled, and hence the first uniformity metric accounts for low values. On the other hand, when the number of feasible solutions is high, each solution is sampled only a few times, and therefore the ratio of the most frequent occurrences to the least frequent ones is naturally limited. On average, SB registers the first uniformity metric 36% more efficiently than the modified IPFP.

Figure 4. Fraction of sampled feasible solutions



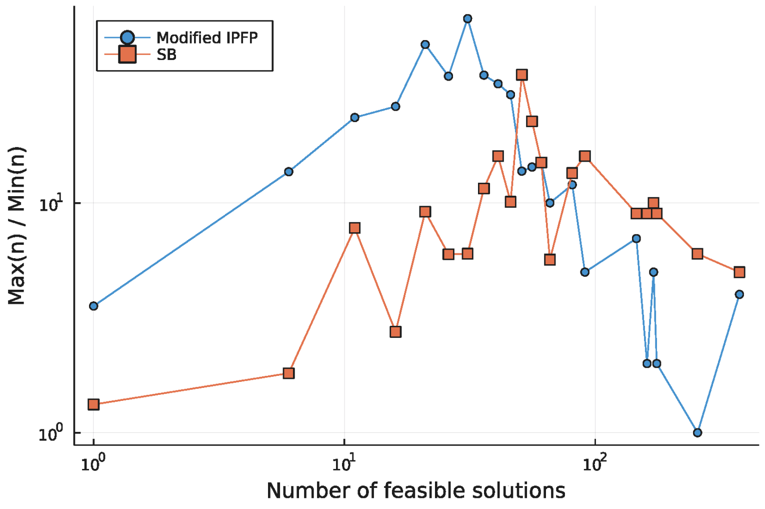
Source: authors' calculations.

Secondly, we test the uniformity with Total Variation Distance δ, defined as in Equation (5) (Levin et al., 2009), where a and b are probability distributions on D (shown in Figure 6). In our case, probability distribution a is an empirical distribution returned by the sampling algorithm, and b is a uniform distribution:

$$\delta(a,b) = \frac{1}{2}\sum_{x\in D}|a(x) - b(x)|. \tag{5}$$

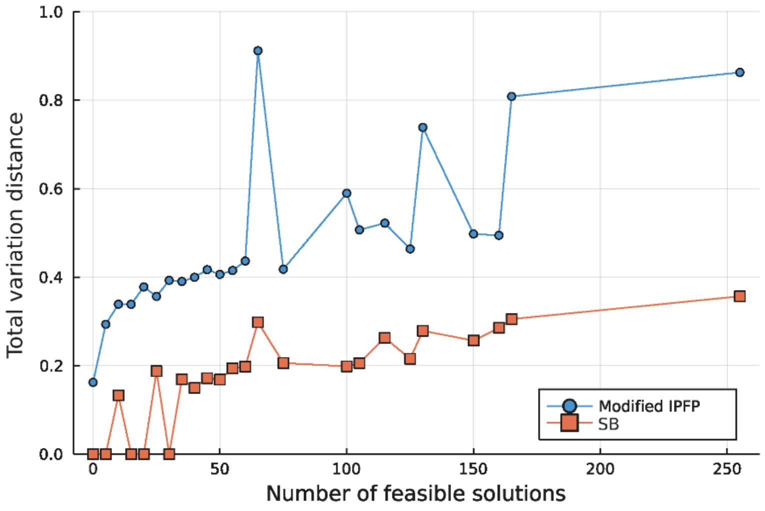
When the uniformity of the sampled solutions is measured with the total variation distance, the SB method outperforms the modified IPFP, regardless of the size of the problem measured with the number of feasible solutions. Although the total variation distance grows for both methods as problems become increasingly more complicated, for SB it never exceeds 0.40, while for the modified IPFP it reaches the level of 0.85.

Figure 5. Uniformity of the sampled solutions – the ratio of the most common solution to the least common solution



Source: authors' calculations.

Figure 6. Uniformity of the sampled solutions – total variation distance



Source: authors' calculations.

4. Conclusions

In this article, we investigated the problem of partially unknown contingency tables. In such arrays, the problem of unknown entries is caused by data privacy requirements, and thus cells with low counts are not reported. We presented our modification of the classic IPFP algorithm and proposed a simulation method incorporating the Shake-and-Bake algorithm. In addition to these methods, we developed a list of heuristics and dimensionality-reduction techniques which, if applied first, simplify the problem and search for all trivial fillings. We conducted a series of experiments to compare both methods in terms of their ability to effectively search through the space of feasible solutions and time complexity.

Our results show that with the increasing dimensionality of the contingency table, the probability that simple heuristics could solve all missing entries rises. In the case of moderately-sized problems, pre-processing required on average 2.3–2.6 rounds to find a solution. Wherever pre-processing was not able to solve the contingency table, we used two competitive methods. In terms of time complexity, our results show that SB outperforms the modified IPFP algorithm when solving smaller problems (of the number of missing entries lower or equal to 150). On average, the time required for SB to sample 1,000 solutions was lower by 64% than the time required for the modified IPFP to do the same. In terms of the ability to search through the space of solutions, SB was able to find 85% of the feasible solutions, while the modified IPFP was able to locate 78%. Moreover, SB samples are characterised by a greater uniformity, which was proven by two different metrics: the ratio of the most frequent solution to the least frequent solution and the total variation distance. We can therefore conclude that SB outperforms the modified IPFP algorithm, as it offers a lower time complexity and a more thorough search of the space of feasible solutions.

In terms of the future development of this study, we plan to improve the functionality of the simulated approach described herein, so as to make possible the sampling from the lattice (Xie et al., 2017). In addition, since the algorithms are designed to solve problems based on sample data, we would like to add an adjustment for population marginal totals (as an optional step).

Acknowledgements

The project has been financed by the Polish National Agency for Academic Exchange. The authors would like to thank Maciej Beręsewicz, PhD, for his substantial help in choosing literature relevant to the presented analysis.

References

Aoki, S., Hara, H., & Takemura, A. (2012). Running Markov Chain Without Markov Bases. In S. Aoki, H. Hara & A. Takemura (Eds.), *Markov Bases in Algebraic Statistics* (pp. 275–286). Springer. https://doi.org/10.1007/978-1-4614-3719-2_16.

Bacharach, M. (1965). Estimating Nonnegative Matrices from Marginal Data. *International Economic Review*, *6*(3), 294–310. https://doi.org/10.2307/2525582.

Bailey, N. T. J. (1995). *Statistical methods in biology* (3rd edition). Cambridge University Press. https://doi.org/10.1017/CBO9781139170840.

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671.

Boender, C. G. E., Caron, R. J., McDonald, J. F., Kan, A. H. G. R., Romeijn, H. E., Smith, R. L., Telgen, J., & Vorst, A. C. F. (1991). Shake-and-Bake Algorithms for Generating Uniform Points on the Boundary of Bounded Polyhedra. *Operations Research*, *39*(6), 945–954. https://doi.org/10.1287/opre.39.6.945.

Bretto, A. (2013). *Hypergraph Theory*. Springer. https://doi.org/10.1007/978-3-319-00080-0.

Csiszár, I. (1975). *I*-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, *3*(1), 146–158. https://doi.org/10.1214/aop/1176996454.

Deming, W. E., & Stephan, F. F. (1940). On a Least Squares Adjustment of a Sampled Frequency Table When the Expected Marginal Totals are Known. *The Annals of Mathematical Statistics*, *11*(4), 427–444. https://doi.org/10.1214/aoms/1177731829.

Diaconis, P., & Sturmfels, B. (1998). Algebraic Algorithms for Sampling from Conditional Distributions. *Annals of Statistics*, *26*(1), 363–397. https://doi.org/10.1214/aos/1030563990.

Dobra, A. (2003). Markov bases for decomposable graphical models. *Bernoulli*, *9*(6), 1093–1108. https://doi.org/10.3150/bj/1072215202.

Dobra, A. (2012). Dynamic Markov Bases. *Journal of Computational and Graphical Statistics*, *21*(2), 496–517. https://doi.org/10.1080/10618600.2012.663285.

Dobra, A., & Fienberg, S. E. (2010). The Generalized Shuttle Algorithm. In P. Gibilisco, E. Riccomagno, M. P. Rogantin & H. P. Wynn (Eds.), *Algebraic and Geometric Methods in Statistics* (pp. 135–156). Cambridge University Press. https://doi.org/10.1017/CBO9780511642401.

Eastman, S. T., & Ferguson, D. A. (2012). *Media Programming: Strategies and Practice* (9th edition). Wadsworth, Cengage Learning.

Facebook.com. (n.d.). *Facebook lookalike audiences*. Retrieved January 23, 2021, from https://www.facebook.com/business/a/custom-to-lookalike-audiences.

Fienberg, S. E. (1970). An Iterative Procedure for Estimation in Contingency Tables. *The Annals of Mathematical Statistics*, *41*(3), 907–917. https://doi.org/10.1214/aoms/1177696968.

Google.com. (n.d.). *About similar segments on the Display Network*. Retrieved January 23, 2021, from https://support.google.com/google-ads/answer/2676774?hl=en.

Kot, M., & Kamiński, B. (2021). Agent Based Model of Cross Media Reach of Advertising. In P. Ahrweiler & M. Neumann (Eds.), *Advances in Social Simulation: Proceedings of the 15th Social Simulation Conference: 23–27 September 2019* (pp. 45–57). Springer. https://doi.org/10.1007/978-3-030-61503-1_5.

Kroese, D. P., Taimre, T., & Botev, Z. I. (2011). *Handbook of Monte Carlo Methods*. John Wiley & Sons. https://doi.org/10.1002/9781118014967.

Levin, D. A., Peres, Y., & Wilmer, E. L. (2009). *Markov Chains and Mixing Times* (2nd edition). American Mathematical Society. http://dx.doi.org/10.1090/mbk/058.

Lovelace, R., Birkin, M., Ballas, D., & van Leeuwen, E. (2015). Evaluating the performance of Iterative Proportional Fitting for spatial microsimulation: new tests for an established technique. *Journal of Artificial Societies and Social Simulation*, *18*(2), 1–25. http://dx.doi.org/%2010.18564/jasss.2768.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., & Tack, G. (2007). MiniZinc: Towards a Standard CP Modelling Language. In C. Bessière (Ed.), *Principles and Practice of Constraint Programming – CP 2007* (pp. 529–543). Springer. https://doi.org/10.1007/978-3-540-74970-7_38.

Payne, G., & Payne, J. (2011). *Key Concepts in Social Research*. SAGE Publications. https://dx.doi.org/10.4135/9781849209397.

R Core Team. (2018). *R: A language and environment for statistical computing*. Retrieved January 23, 2021, from https://www.R-project.org.

Slavković, A. B. (2010). Partial Information Releases for Confidential Contingency Table Entries: Present and Future Research Efforts. *Journal of Privacy and Confidentiality*, *1*(2), 253–264. https://doi.org/10.29012/jpc.v1i2.577.

Slavković, A. B., & Lee, J. (2010). Synthetic two-way contingency tables that preserve conditional frequencies. *Statistical Methodology*, *7*(3), 225–239. https://doi.org/10.1016/j.stamet.2009.11.002.

van Valkenhoef, G., & Tervonen, T. (2019, January 8). *'Hit and Run' and 'Shake and Bake' for Sampling Uniformly from Convex Shapes*. https://cran.r-project.org/web/packages/hitandrun/hitandrun.pdf.

Xie, C., Zhong, W., & Mueller, K. (2017). A Visual Analytics Approach for Categorical Joint Distribution Reconstruction from Marginal Projections. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 51–60. https://doi.org/10.1109/TVCG.2016.2598479.

Zelterman, D., & Louis, T. A. (2019). Contingency Tables in Medical Studies. In J. C. Bailar & F. Mosteller (Eds.), *Medical Uses of Statistics* (2nd edition; pp. 293–310). CRC Press. https://doi.org/10.1201/9780429187445.