DIFFUSR: DISTORTION-FREE SWAP-RANDOMIZATION FOR STATISTICALLY-TESTING DATA MINING RESULTS

Submitted to the Department of Computer Science of Amherst College

in partial fulfillment of the requirements

for the degree of Bachelor of Arts with honors

April 11, 2022

Author: Alexander Lee

Advisor: Prof. Matteo Riondato

Abstract

We present DiFfuSR, a set of algorithms for drawing transactional datasets from a null model, to be used for the statistical validation of knowledge discovery results. Our algorithms assume the widely-adopted null model that maintains transaction lengths and item supports, and they are based on swap randomization, i.e., randomly swapping items between transactions, in a Markov-Chain Monte-Carlo (MCMC) setting, using Metropolis-Hastings. In contrast with previous work, our approach does not suffer from the distortion of the null space introduced by identifying the set of transactional datasets with the set of binary matrices, which could lead to unknown false positives in the output. DiFfuSR uses carefully-designed Markov chains with appropriate stationary distributions to sample datasets from the null model. The results of our experimental evaluation show that DiFfuSR solves the distortion issue, is fast and (empirically) rapidly-mixing, and scales well as the dataset size grows.

Acknowledgements

I am deeply grateful to my advisor Prof. Matteo Riondato. I remember working as his teaching assistant in my junior year and asking if I could do research with him. To my surprise, he said yes without hesitation. Ever since then, he has been there to support me. When I had rough weeks and showed up with nothing done, he was patient and understanding. When things were not working as planned, he managed to instill positivity. When I made mistakes, he continued to place his trust in me. I am always so inspired by his brilliance and passion for research that I cannot help but leave our meetings with excitement and a smile. Because of him, I am now more confident in my abilities. He has taught me so much, perhaps more than he has realized. Most notably, he has taught me what outstanding mentorship looks like. Matteo has worked tirelessly to help me. And I know he has done the same for so many around me. In fact, I am confident he will continue to do so for the students that follow.

I would also like to give my sincere thanks to the computer science faculty, particularly to those who taught me: Prof. Scott Alfeld, Prof. Kristen Gardner, Prof. Scott Kaplan, Prof. Lyle McGeoch, Prof. Will Rosenbaum, and Prof. Lee Spector. The department's professors have my utmost respect for continuing to do such a phenomenal job in the face of unprecedented circumstances. I would have never been able to write a thesis without all that I have learned from them.

After my thesis talk, Prof. Lee Spector posed the question of whether distortion is always undesirable. This question helped bring forth the realization that some crucial ideas of this work were not emphasized enough. I hope these ideas are now clear, and I would like to express my appreciation.

Having worked on a similar research topic, Stefan Walzer-Goldfeld and Steedman Jenkins helped me get up to speed in the beginning and provided feedback on the thesis later on. Their contributions were imperative, and I could not have asked for better partners.

Lee Jiaen, Holden Lee, and Sarah Park collaborated with me on multiple computer science assignments. There were times when we hit our heads against the wall,[1] but there were also times when we shouted in celebration after things finally worked. They have been part of what makes computer science fun.

Lesley Zheng was like another older sister to me. She was one of my teaching assistants when I started taking computer science courses. Since then, she has always been happy to help me, whether with the major or my future career. I hope that I can return the favor someday.

Friends from the Cohan suite, the Greenway suites, and my international family have always managed to make me laugh and bring joy into my life. Each one of them has made Amherst more memorable, and for that, I am so blessed. I know they will all achieve great things, and I will miss them a lot.

I cannot talk about my time at Amherst without mentioning Sona. She has been so good to me. Whenever I felt overwhelmed, she gave me comfort. Whenever I was working too much, she forced me to live a little. I only hope that she feels supported too, as I am so grateful for everything she has done.

[1] Mostly figuratively. . .

Jason Liu, Hiro Wang, Joanne Wang, and Sean Wang grew up with me in Beijing, and we moved to the U.S. for college around the same time. Even though I have not been able to return to Beijing in a while, seeing them always brought a sense of home and comfort. I am so thankful for our friendship.

Ariana overlapped with me at Amherst for two years. I am so lucky to have her as an older sister since I would be completely lost without her. I somehow manage to follow her wherever she goes, so while we are living apart now, I have a feeling we will be with each other soon.

I am greatly indebted to my parents. They have continually given me so much love and support from across the world, even during times when I took them for granted. I hope one day I can give back to them even half of what I have received, as they have given me so much.

Contents

Chapter 1

Introduction

Knowledge discovery results obtained from a dataset (e.g., patterns, clusters, anomalies) should be *statistically validated* in order to ensure that they are not just due to the randomness in the data-generating process [Gionis et al., 2007, Hämäläinen and Webb, 2019, Pellegrina et al., 2019b]: the goal of the analysis is to *gain knowledge of this process through the observed dataset*, rather than knowledge of the dataset itself.

One such validation subjects the results to *statistical hypothesis tests* [Wasserman, 2005, Ch. 10] (see Sect. 2.1): results that pass the tests are marked as *(statistically) significant*, as there is evidence that they give information on the data generating process.

The significance of the results is assessed against a *null model* (see Sect. 2.1), which is a collection of *possible* datasets that the unknown process *may* generate, and a probability distribution over this collection. The null model captures, in some sense, what is assumed or already known about the data generating process, and the results are assessed against it to understand in what way they cannot be explained by the existing knowledge or assumptions.

The choice of the null model made by the user must be *deliberate and informed*, as the meaning of "statistically significant" depends on the null model. For example, the results deemed significant under one null model cannot in general be compared to those deemed significant under a different null model. Nevertheless, as the aphorism says: "all models are wrong, but some are useful" (George E. P. Box), and some null models may be more appropriate for testing the significance of the results than others, because they more closely represent the settings of the task. It is therefore important that users have access to a

variety of null models and are informed of the differences between them, in order to be able to select the more appropriate one for their needs. In this work, we introduce a null model for testing the significance of results obtained from *binary transactional datasets*, such as (but not limited to) collections of frequent itemsets [Agrawal and Srikant, 1994]. A null model is, in some sense, independent from the task whose results one wants to validate, as a null model is a set of datasets, not results, but is used to evaluate results. Our model is more appropriate for evaluating the results of common tasks on transactional datasets than existing ones [Gionis et al., 2007], as we explain in the *Contributions* paragraph and in Sect. 3.1.

Many statistical hypothesis tests are based on *resampling* [Westfall and Young, 1993]: they analyze multiple datasets drawn from the null model distribution, in order to approximate the distribution of the test statistic and compare its value in the observed data against such distribution. Thus, it is necessary to develop computationally-efficient procedures for sampling such datasets from the null model distribution.

1.1 Contributions

We study the problem of evaluating the significance of results extracted from unlabeled transactional datasets using resampling-based statistical hypothesis tests.

- We introduce a novel null model that preserves transaction lengths and item supports, but does not suffer from the *distortion* introduced by equating the set of datasets and the set of binary matrices (see Sect. 1.2 and 3.1), and is therefore more appropriate for evaluating the results of knowledge discovery tasks on transactional datasets. The user can specify *any* distribution over the set of datasets.

- We present DiFfuSR (for *Di*stortion-*F*ree *S*wap *R*andomization), a set of algorithms for sampling datasets from the null model we introduce. These algorithms are based on *swap-randomization* (see Sect. 3.2) and take a Markov-Chain Monte-Carlo (MCMC) approach on appropriately-defined Markov chains (see Sect. 2.2) over either the set of binary matrices (for our "naïve" algorithm), or the set of datasets (for our "refined" algorithm), whose stationary distribution is the user-specified sampling distribution.

- Our algorithms can be seen as the core of procedures to assess the statistical validity of data mining

results using a resampling approach [Westfall and Young, 1993] (see Sect. 2.1). To show the usefulness of our algorithms, we focus on the task of evaluating the significance of the number of frequent itemsets in a dataset, by computing an empirical p-value of this statistic.

- The results of our experimental evaluation show that the distortion issue is very relevant and affects the outcome of the testing of data mining results. DiFFuSR does not suffer from it, while being fast, (empirically) rapidly-mixing, and scalable as the dataset grows.

1.2 Related Work

Evaluating the statistical significance of results obtained by mining transactional datasets has been recognized as important soon after the first efficient mining algorithms were introduced. For example, Brin et al. [1997] use the chi-square test to assess the significance of association rules, while Megiddo and Srikant [1998] use statistical significance testing to measure the number of false discoveries.

Many measures of interestingness, both for individual patterns and for sets of patterns [Kontonasios et al., 2012, Tan et al., 2004, Vreeken and Tatti, 2014], were introduced with the goal of mitigating the pattern explosion observed when using frequency as a proxy of interestingness, but they do not offer the strict statistical guarantees that allow precise statements about the interestingness of a pattern w.r.t. the data generation process.

The framework of statistical hypothesis testing [Wasserman, 2005, Ch. 10] offers such guarantees, and is now the de facto standard taken by works in statistically-significant pattern mining [Hämäläinen and Webb, 2019, Pellegrina et al., 2019b]. The area of statistical pattern mining can be split into two classes, depending on whether each transaction in the dataset is associated to a (binary or real) label. Our method DiFFuSR works on *unlabeled* datasets thus it can be used for mining significant patterns in this setting, but many works studied the labeled case [Llinares-López et al., 2015, Minato et al., 2014, Pellegrina and Vandin, 2020, Pellegrina et al., 2019a, Terada et al., 2013, Wu et al., 2016, Komiyama et al., 2017]. To keep the presentation focused, we only discuss contributions for unlabeled datasets. The tutorial by Pellegrina et al. [2019b] offers an in-depth presentation of the labeled case.

Mining significant patterns requires *testing multiple hypotheses*, one per pattern. Due to the random-

ness of the data generation process, and the fact that one only has access to the observed dataset, it is necessary to accept the possibility of wrongly marking a pattern as significant even if it is not, i.e., of making a *false discovery* involving that pattern, or, equivalently, that the pattern is a *false positive*. Algorithms for significant pattern mining must therefore give *probabilistic guarantees* on the presence of false positives in their output. The most common of such guarantees is a bound on the *Family-Wise Error Rate (FWER)*, i.e., on the probability that the output collection contains *any* false positives. In pattern mining, Webb [2006, 2007], Hämäläinen [2010], Low-Kam et al. [2013], and many others (see the tutorial by Hämäläinen and Webb [2019]) present methods to control the FWER using the *Bonferroni correction* [Bonferroni, 1936]. This approach, even when refined [Holm, 1979, Webb, 2008], does not take into account dependencies and correlations among hypotheses, i.e., the *structure* of the hypothesis class, which is very rich for the classes associated to patterns. Additionally, these classes contain many patterns that would never be marked as significant (e.g., because they do not appear in the observed dataset), thus using the Bonferroni correction, while controlling the FWER to the desired level, results in low statistical power, i.e., in not marking many true significant patterns as such. One alternative is to control the False Discovery Rate (FDR), i.e., the expected number of false positives, as proposed by Kirsch et al. [2012]. The FDR is a less stringent guarantee than the FWER, thus is only appropriate in some settings, while controlling the FWER is required in many cases. Resampling methods [Westfall and Young, 1993] take into account the structure of the hypothesis class and offer high statistical power, while still bounding the FWER. DiFfuSR is designed to be used in such methods.

The resampling method by Gionis et al. [2007], while it can be used as the core component of a significant pattern mining algorithm, it is, like our work, more generic and useful for assessing the validity of data mining results. They do not discuss controlling the FWER, a void filled by Hanhijärvi [2011]. Gionis et al. [2007] look at transactional datasets as binary matrices, and consider the null model composed by *all* such matrices with the same row and column sums (a.k.a., margins) as the observed dataset. We discuss Gionis et al. [2007]'s method in detail in Sect. 3.1. The issue with this approach that we identify and solve is that the space of matrices with constrained margins does not actually have a one-to-one correspondence with the space of transactional datasets with the same transaction lengths and item frequencies, thus creating a *distortion* of the latter space, and effectively leading to performing statistical tests w.r.t. a *different*

null model. The statistical significance of the observed results depends on the null model, thus the distortion may make results that are not actually significant appear as such, and vice versa. Our algorithms are *distortion-free* and sample w.r.t. the intended null model.

The randomization approach by Gionis et al. [2007] has been extended to other types of data (database tables [Ojala et al., 2010], real-valued and mixed-value matrices [Ojala et al., 2008, Ojala, 2010], graphs [Hanhijärvi et al., 2009, Günnemann et al., 2012], genomic data [Ferkingstad et al., 2015]), to different kinds of patterns (sequences [Tonon and Vandin, 2019], subgroups [Duivesteijn and Knobbe, 2011]), and to the iterative setting [Hanhijärvi et al., 2009]. The distortion issue affecting Gionis et al. [2007]'s approach is also present in many of these methods, and our proposed solution can be adapted to these cases. We discuss some of them in Sect. 3.3, though an in-depth treatment is an interesting direction for future work.

Chapter 2

Preliminaries

We now introduce the main concepts used in the work. To exemplify the application of DiFfuSR, we focus on the task of evaluating whether the size of the collection of frequent itemsets in an observed dataset is statistically significant. We choose this task because it allows for a self-contained presentation that is accessible also to non-experts, rather than describing an arguably more interesting, but certainly more convoluted task such as mining statistically-significant frequent patterns. We remark that DiFfuSR can be used to validate any kind of results obtained from transactional datasets, including mining statistically-significant frequent patterns (not necessarily itemsets), evaluating the correlations between different items, and much more (see also Sect. 3.3).

Let $\mathcal{I} \doteq \{a_1, \ldots, a_n\}$ be a finite *alphabet* of $n \doteq |\mathcal{I}|$ *items*. Without loss of generality, we can assume $\mathcal{I} = \{1, \ldots, n\}$. An *itemset* $A \subseteq \mathcal{I}$ is any non-empty subset of $\mathcal{I}$. A *dataset* $\mathcal{D} \doteq \{t_1, \ldots, t_m\}$ is a finite *bag* of $m \doteq |\mathcal{D}|$ itemsets which, as the elements of $\mathcal{D}$, are known as *transactions*. An itemset A *appears* in a transaction t when $A \subseteq t$. The *support* $\sigma_{\mathcal{D}}(A)$ *of itemset* A *in dataset* $\mathcal{D}$ is the number of transactions of $\mathcal{D}$ in which A appears, i.e.,

$$\sigma_{\mathcal{D}}(A) \doteq |\{t \in \mathcal{D} : A \subseteq t\}| \ .$$

Given a *minimum support threshold* $\theta \in [1, |\mathcal{D}|]$, the set $\mathsf{FI}_{\mathcal{D}}(\theta)$ of *frequent itemsets in* $\mathcal{D}$ *w.r.t.* θ is the set of itemsets that have support at least θ in $\mathcal{D}$, i.e.,

$$\mathsf{FI}_{\mathcal{D}}(\theta) \doteq \{A \subseteq \mathcal{I}, A \neq \emptyset \ : \ \sigma_{\mathcal{D}}(A) \geq \theta\} \ .$$

As an example, suppose we have an alphabet $\mathcal{I} = \{1, 2, 3\}$ and a dataset $\mathcal{D} = \{\{1, 2\}, \{1, 3\}, \{3\}\}$. The support $\sigma_{\mathcal{D}}(\{1\})$ of itemset $\{1\}$ in dataset $\mathcal{D}$ equals 2, since itemset $\{1\}$ appears in only two transactions of $\mathcal{D}$: $\{1, 2\}$ and $\{1, 3\}$. By similar reasoning, we also have that $\sigma_{\mathcal{D}}(\{1, 2\}) = 1$. With the same dataset $\mathcal{D}$, the set of frequent itemsets $\mathsf{FI}_{\mathcal{D}}(1)$ in $\mathcal{D}$ w.r.t. a minimum support threshold of 1 equals $\{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}\}$, because each itemset in the set appears at least once in $\mathcal{D}$. Similarly, we also have that $\mathsf{FI}_{\mathcal{D}}(2) = \{\{1\}, \{3\}\}$ and $\mathsf{FI}_{\mathcal{D}}(3) = \emptyset$.

2.1 Null Models and Significant Results

Statistical significance is assessed w.r.t. a user-specified *null model*. A null model is, for our purposes, a pair $\Pi \doteq (\mathcal{Z}, \pi)$, where $\mathcal{Z}$ is a set of datasets, known as the *null set* or *null space*, and π is a probability distribution over $\mathcal{Z}$. In the setting we consider, there is one fixed *observed dataset* $\mathring{\mathcal{D}}$, and the null set $\mathcal{Z}$ is such that $\mathring{\mathcal{D}} \in \mathcal{Z}$ and $\mathcal{Z}$ contains all and only datasets that share some characteristic properties with $\mathring{\mathcal{D}}$. Thus, the null model depends on the observed dataset $\mathring{\mathcal{D}}$. We follow all previous work and only consider null models where $\mathcal{Z}$ contains all and only the datasets $\mathcal{D}$ such that

- $|\mathcal{D}| = |\mathring{\mathcal{D}}| = m$, i.e., $\mathcal{D}$ has the same *size*, i.e., number m of transactions, as $\mathring{\mathcal{D}}$; and
- if we let $\mathcal{D} = \{t_1, \ldots, t_m\}$ and $\mathring{\mathcal{D}} = \{\mathring{t}_1, \ldots, \mathring{t}_m\}$, there is a *permutation* ρ of $\{1, \ldots, m\}$ such that $|t_i| = \left|\mathring{t}_{\rho(i)}\right|$ for $1 \leq i \leq m$, i.e., $\mathcal{D}$ has the same *distribution of transaction lengths* as $\mathring{\mathcal{D}}$; and
- $\sigma_{\mathcal{D}}(\{a\}) = \sigma_{\mathring{\mathcal{D}}}(\{a\})$, for every *item* $a \in \mathcal{I}$, i.e., each item has the same support in $\mathcal{D}$ and $\mathring{\mathcal{D}}$.

We call such null models "Size, Length and Item-Supports Preserving" (SLISP). Given $\mathring{\mathcal{D}}$, all SLISP null models for $\mathring{\mathcal{D}}$ have *the same null set* $\mathcal{Z}$, i.e., they differ only in the distribution π over $\mathcal{Z}$. The user may specify any distribution π over $\mathcal{Z}$, although previous work mostly focused on π being the uniform distribution over $\mathcal{Z}$.

Continuing our running example, suppose we observe dataset $\mathring{\mathcal{D}} = \{\{1, 2\}, \{1, 3\}, \{3\}\}$. The following are three examples of datasets in the null set $\mathcal{Z}$ (which also includes other datasets)

$$\{\{1, 2\}, \{1, 3\}, \{3\}\},\ \{\{1, 3\}, \{1, 3\}, \{2\}\},\ \text{and } \{\{2, 3\}, \{1, 3\}, \{1\}\}.$$

Each of these datasets preserve the size, distribution of transaction lengths, and item supports of the observed dataset $\mathring{\mathcal{D}}$. The observed dataset $\mathring{\mathcal{D}}$ itself is in the null set $\mathcal{Z}$.

When using statistical hypothesis testing to assess the validity of knowledge discovery results, one is interested in understanding how "typical" the results from $\mathring{\mathcal{D}}$ are with respect to the distribution of the results from datasets sampled from the null model Π: if they are not "typical", the results are considered *significant* (under the assumed null model). For example, if we want to assess whether the number $|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|$ of frequent itemsets w.r.t. θ in $\mathring{\mathcal{D}}$ is significant, we could make the *null hypothesis*

$$H_0 \doteq \text{“}|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)| = \mathbb{E}_{\mathcal{D}\sim\pi}[|\mathsf{FI}_{\mathcal{D}}(\theta)|]\text{”}, \tag{2.1}$$

and then perform a *statistical hypothesis test* to assess whether there is evidence that this null hypothesis is true. If it is not, we *reject* it and say that the value $|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|$ appears significant.

One way to perform such a test is to approximate the distribution of the statistic of interest (in this case, the number of frequent itemsets) by *sampling datasets from the null model* [Westfall and Young, 1993], and then compare the observed statistic $|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|$ to the (empirical) distribution, as follows. Assume to sample a collection $\mathcal{T} \doteq \{\mathcal{D}_1, \ldots, \mathcal{D}_\ell\}$ of ℓ datasets independently from $\mathcal{Z}$ according to π. The *(empirical)* p-value $\tilde{\mathsf{p}}(\mathring{\mathcal{D}}, \mathcal{T})$ is defined as the fraction of datasets in $\mathcal{T} \cup \{\mathring{\mathcal{D}}\}$ with a number of frequent itemsets w.r.t. θ that is not smaller than the one observed in $\mathring{\mathcal{D}}$, i.e.,

$$\tilde{\mathsf{p}}(\mathring{\mathcal{D}}, \mathcal{T}) \doteq \frac{1 + |\{1 \leq i \leq \ell : |\mathsf{FI}_{\mathcal{D}_i}(\theta)| \geq |\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|\}|}{1 + \ell} \ .$$

Let now $\alpha \in (0, 1)$ be a user-specified *acceptable probability of error*. If $\tilde{\mathsf{p}}(\mathring{\mathcal{D}}, \mathcal{T}) \leq \alpha$, then we say that $|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|$ is significant at level α, which can interpreted as meaning that there is evidence that the null hypothesis from (2.1) is false and should be rejected. The value α is the probability of committing a *false discovery*, i.e., of wrongly declaring the observed results significant when they are not.

In the situation we just considered, there was a single null hypothesis, but it is often the case that *multiple* hypotheses must be tested. For example, in significant pattern mining (see Sect. 1.2), there is one hypothesis per pattern. In this case, one wants guarantees on the *Family-Wise Error Rate* (FWER), i.e., on the probability of making *any* false discovery. To ensure that the FWER is bounded by an user-specified

threshold $\delta \in (0, 1)$, the p-value of each hypothesis to be tested is compared to an *adjusted critical value* $\alpha(\Pi, \mathcal{H}, \delta)$, where $\mathcal{H}$ is the set of the null hypotheses of interest. For example, the classic Bonferroni correction [Bonferroni, 1936] uses $\delta/|\mathcal{H}|$ as the adjusted critical value. We discussed the limitations of this approach in Sect. 1.2. *Resampling approaches* [Westfall and Young, 1993] solve those issues by computing adjusted critical values using *datasets sampled* according to π, and they have been used with success in significant itemset mining [Pellegrina et al., 2019b].

Regardless of how many hypotheses are tested, this discussion shows that *efficient* procedures to draw datasets from $\mathcal{Z}$ independently according to π are needed for assessing the statistical validity of results obtained from these datasets. Developing such procedures is the goal of our work.

2.2 Markov-Chain-Monte-Carlo and Metropolis-Hastings

All sampling procedures we discuss follow the *Markov-Chain-Monte-Carlo* (MCMC) approach using the *Metropolis-Hastings (MH) algorithm*. We give here a description *tailored to our needs*, and refer the reader to [Mitzenmacher and Upfal, 2005, Ch. 7 and 10] for a general discussion.

A (finite, time-homogeneous) Markov chain is a sequence $(X_0, X_1, X_2, \dots)$ of random variables taking values over a (finite) set $\mathcal{S}$, such that the *state* X_t at *time step* $t \in \mathbb{N}$ depends *only* on the state X_{t-1} at time $t-1$. W.l.o.g., we can assume that $\mathcal{S} = \{0, 1, 2, \dots, |\mathcal{S}|\}$. For any two states $i, j \in \mathcal{S}$, the *transition probability* $P_{i,j}$ is the conditional probability that $X_t = j$ given that $X_{t-1} = i$, for any time step t. The transition matrix $\mathbf{P}$ is the $|\mathcal{S}| \times |\mathcal{S}|$ matrix whose (i, j) entry is $P_{i,j}$. Given $\mathbf{P}$, we can define a directed weighted graph $G = (\mathcal{S}, E, \mathsf{w})$, where $(i, j) \in E$ iff $P_{i,j} > 0$. The weight $\mathsf{w}(i, j)$ for any $(i, j) \in E$ is $P_{i,j}$. If the neighborhood structure (i.e., E) and the transition matrix $\mathbf{P}$ have some specific properties,[1] then there is one and only one ζ such that $\zeta = \zeta\mathbf{P}$. The vector ζ is a probability distribution over $\mathcal{S}$ known as the (unique) *stationary distribution* of the Markov chain. It represents the distribution of the state X_t as $t \to \infty$.

Here is the idea behind MCMC sampling with MH. Suppose that we want to sample from a population $\mathcal{P}$, according to a distribution ξ over $\mathcal{P}$. MH designs a Markov chain with state space $\mathcal{S} = \mathcal{P}$ and stationary

[1] Formally, if the chain is *irreducible and aperiodic* [Mitzenmacher and Upfal, 2005, Thm. 7.7]. All Markov chains we consider have these properties.

distribution $\zeta = \xi$. Let t be a time step sufficient for the distribution of the state of the chain to be the stationary distribution. Then, X_t is a sample of $\mathcal{P}$ according to ξ.

To use MCMC sampling with MH, one must first specify an initial neighborhood structure for $\mathcal{P}$, which is usually "naturally" imposed by properties of $\mathcal{P}$: if one element $a \in \mathcal{P}$ can be obtained by a "simple modification" to another element b, there should likely be an edge from b to a in the weighted directed graph representing the Markov chain. The definition of "simple modification" depends on the settings. Additionally, for each element $a \in \mathcal{P}$, one must specify a probability distribution η_a over the out-neighbors of a. MH uses the specified neighborhood structure, with the addition of *self-loops* from each $a \in \mathcal{P}$ to itself, if not already present. It gives a way to (implicitly) set the transition probabilities so that the stationary distribution ζ of the resulting Markov chain is the desired sampling distribution ξ. Specifically, at each step t, the next state X_{t+1} is chosen in two phases. First, a *(out-) neighbor* b of $X_t = a$ is chosen by sampling according to the distribution η_a over the out-neighbors of a. Then, X_{t+1} is set to b with probability

$$\min\left\{1, \frac{\xi(b)\eta_b(a)}{\xi(a)\eta_a(b)}\right\}, \tag{2.2}$$

otherwise $X_{t+1} = a$.

Thus, the necessary "ingredients" to use MH are the population $\mathcal{P}$, the sampling distribution ξ, the initial neighbor structure, and the distribution η_a for every $a \in \mathcal{P}$. In the next chapter, we describe algorithms for sampling datasets from SLISP null models and for each of them we define these ingredients.

Chapter 3

Sampling Datasets

3.1 Existing Method

Gionis et al. [2007] introduce an algorithm (which we refer to as GMMT) for sampling *uniformly* at random from a set $\mathcal{M}$ of *binary matrices*. For ease of presentation in this subsection, we limit ourselves to the uniform distribution over $\mathcal{M}$, but all that we say here can be extended to a generic distribution over $\mathcal{M}$. The basis for presenting GMMT as an algorithm for sampling *datasets* from a SLISP null model $\Pi = (\mathcal{Z}, \mathcal{U}(\mathcal{Z}))$ for an observed dataset $\mathring{\mathcal{D}}$ is the *unstated assumption* that there is *always* a *bijection* between $\mathcal{Z}$ and the set $\mathcal{M}$ of binary matrices from which GMMT samples. We show (Lemma 3.2) that *the assumption is false*, and GMMT samples datasets from a "distorted" null model with a probability distribution that is not necessarily uniform.

3.1.1 Datasets as Matrices

We first must give some more detail about GMMT. This algorithm implicitly assumes that datasets, rather than being *bags* of transactions, are *sequences*, i.e., the transactions in them have a fixed *order*. We refer to datasets seen as sequences with the term "seq-datasets". Clearly, since there are many possible orderings of the transactions of a dataset, multiple seq-datasets may correspond to the same dataset. This assumption allows to define a bijection that maps any seq-dataset $\mathcal{D} = \langle t_1, \ldots, t_m \rangle$ with m transactions built on the alphabet $\mathcal{I} = \{1, \ldots, n\}$ of items to one and only one binary matrix $M_{\mathcal{D}} \in \{0,1\}^{m \times n}$ with m rows and

n columns by setting the entry $M_{\mathcal{D}}(i,j)$ to 1 iff $j \in t_i$, $1 \leq j \leq n$, $1 \leq i \leq m$. For example, assuming $\mathcal{I} = \{1,2,3\}$, the seq-dataset

$$\mathcal{D}' = \langle \{1,2\}, \{1,3\}, \{3\} \rangle \tag{3.1}$$

is mapped to the matrix

$$M_{\mathcal{D}'} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} . \tag{3.2}$$

Given an observed dataset $\mathring{\mathcal{D}}$, GMMT maps it to a seq-dataset $\mathring{\mathcal{D}}' = \langle t_1, \ldots, t_m \rangle$ by arbitrarily choosing an order for the transactions in $\mathring{\mathcal{D}}$. GMMT then defines the set $\mathcal{M} \subseteq \{0,1\}^{m \times n}$ of all $m \times n$ binary matrices M such that

- for $1 \leq i \leq m$, $r_i \doteq \sum_{j=1}^{n} M(i,j) = |t_i|$, i.e., the *row-sum* r_i (that is, the number of 1's in row i) equals the length of the transaction t_i; and
- for $1 \leq j \leq n$, $c_j \doteq \sum_{i=1}^{m} M(i,j) = \sigma_{\mathring{\mathcal{D}}}(\{j\})$, i.e., the *column-sum* c_j (that is, the number of 1's in column j) equals the support of item j in $\mathring{\mathcal{D}}$.

The set $\mathcal{M}$ depends on both $\mathring{\mathcal{D}}$ and the chosen ordering of its transactions to obtain the seq-dataset $\mathring{\mathcal{D}}'$.

Again, suppose for example we observe dataset $\mathring{\mathcal{D}} = \{\{1,2\}, \{1,3\}, \{3\}\}$. Further suppose that the chosen ordering of the transactions of $\mathring{\mathcal{D}}$ is such that we obtain $\mathring{\mathcal{D}}'$ as the seq-dataset $\mathcal{D}'$ from (3.1). It follows that the matrix representation $M_{\mathring{\mathcal{D}}'}$ of the seq-dataset $\mathring{\mathcal{D}}'$ is the matrix $M_{\mathcal{D}'}$ from (3.2). Then, matrices M in the set $\mathcal{M}$ of binary matrices include (but are not limited to)

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ and } \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

since each matrix M preserves the row-sums and column-sums of the matrix $M_{\mathring{\mathcal{D}}'}$. Notice that the matrix $M_{\mathring{\mathcal{D}}'}$ itself is in the set $\mathcal{M}$ of binary matrices. Also observe that the first two rows of the first matrix $M_{\mathring{\mathcal{D}}'}$ can be reordered to obtain the seconds matrix, while the rows cannot be reordered to obtain the third matrix but this matrix is still in $\mathcal{M}$. This observation will be important in the proof for Lemma 3.2.

3.1.2 Swap Randomization

GMMT samples from $\mathcal{M}$ uniformly at random using the MCMC approach with MH (Sect. 2.2). The state space is $\mathcal{M}$ and there is an edge from matrix M' to matrix M'' if there are two row indices $1 \leq r_1, r_2 \leq m$ and two column indices $1 \leq c_1, c_2 \leq n$ such that $M'(r_1, c_1) = 1$, $M'(r_1, c_2) = 0$, $M'(r_2, c_1) = 0$, $M'(r_2, c_2) = 1$, and M'' can be obtained from M' by setting $M'(r_1, c_1) = 0$, $M'(r_1, c_2) = 1$, $M'(r_2, c_1) = 1$, and $M'(r_2, c_2) = 0$, i.e., by performing a single *swap*. For example, given a matrix

$$M' = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \in \mathcal{M},$$

a different matrix

$$M'' = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \in \mathcal{M}$$

can be obtained by performing a single swap w.r.t. to row indices $r_1 = 1, r_2 = 3$ and column indices $c_1 = 2, c_2 = 3$. The distribution η_M is the uniform over the out-neighbors of $M \in \mathcal{M}$, i.e., $\eta_M(M') \doteq 1/\mathsf{mts}(M)$, where $\mathsf{mts}(M)$ of is the number of out-neighbors of M. Gionis et al. [2007, Alg. 2, Thm. 4.3] give procedures to draw a neighbor according to η_M, and to compute $\mathsf{mts}(M)$, $M \in \mathcal{M}$, which are needed by MH. We use this result in our algorithms, so we report it here for self-containedness.

Theorem 3.1 ([Gionis et al., 2007, Thm. 4.3])**.** *Let $M_{\mathcal{D}}$ be a matrix representation of dataset $\mathcal{D}$, and let $Q = M_{\mathcal{D}} M_{\mathcal{D}}^{\mathsf{T}}$. Then*

$$\mathsf{mts}(M_{\mathcal{D}}) = \mathsf{J}(\mathcal{D}) - \mathsf{Z}(\mathcal{D}) + \mathsf{K}(M_{\mathcal{D}}), \tag{3.3}$$

where

$$\mathsf{J}(\mathcal{D}) \doteq \frac{1}{2}\left(\left(1+\sum_{t\in\mathcal{D}}|t|\right)\sum_{t\in\mathcal{D}}|t| - \sum_{t\in\mathcal{D}}|t|^2 - \sum_{i\in\mathcal{I}}\left(\sigma_{\mathcal{D}}(\{i\})\right)^2\right),$$

$$\mathsf{Z}(\mathcal{D}) \doteq \sum_{t\in\mathcal{D}}\sum_{i\in t}[(|t|-1)(\sigma_{\mathcal{D}}(\{i\})-1)], \text{ and}$$

$$\mathsf{K}(M_{\mathcal{D}}) \doteq \sum_{k=1}^{m}\sum_{\ell=1}^{k-1}[(Q(\ell,k))^2 - Q(\ell,k)] \ .$$

A run of the chain can be seen as a sequence of swaps, hence the name *swap randomization* for the approach taken by GMMT. Define the surjective function $\mathsf{dat}(\cdot)$ from $\mathcal{M}$ to the null set $\mathcal{Z}$ of the SLISP null model for $\mathring{\mathcal{D}}$, which maps $M \in \mathcal{M}$ to the *unique* dataset obtained by considering as a bag the seq-dataset mapped to M. After a sufficient number of swaps, the state M of the chain is distributed uniformly *among* $\mathcal{M}$, and the dataset $\mathsf{dat}(M)$ is returned as a sample from $\mathcal{Z}$.

Lemma 3.2. *There exists an observed dataset $\mathring{\mathcal{D}}$ such that, if we let M be a matrix drawn uniformly at random from $\mathcal{M}$, then $\mathsf{dat}(M)$ is not chosen uniformly at random from $\mathcal{Z}$.*

Proof. Let $\mathring{\mathcal{D}} = \{\{1,2\},\{1,3\},\{3\}\}$, and assume, w.l.o.g., that $\mathring{\mathcal{D}}'$ is the seq-dataset $\mathcal{D}'$ from (3.1), then clearly the matrix $M_{\mathcal{D}'}$ from (3.2) belongs to $\mathcal{M}$.

The set $\mathcal{M}$ also contains the matrix M'' obtained by reordering the first two rows of $M_{\mathcal{D}'}$ from (3.2), which corresponds to the seq-dataset $\mathcal{D}'' = \langle\{1,3\},\{1,2\},\{3\}\rangle$. The seq-datasets $\mathcal{D}'$ from (3.1) and $\mathcal{D}''$ are *different*, but their corresponding datasets respectively are *the same dataset*, i.e., $\mathsf{dat}(M_{\mathcal{D}'}) = \mathsf{dat}(M'')$. Thus, the set $\mathcal{M}$ contains *at least two* matrices mapped to seq-datasets that, when seen as bags, are equal to $\mathring{\mathcal{D}}$. From the definition of $\mathcal{M}$, it holds that $\mathcal{M}$ also contains

$$A \doteq \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \ .$$

It holds that $\mathsf{dat}(A) = \{\{1,3\},\{1,3\},\{2\}\}$. Again from the definition of $\mathcal{M}$, it holds that there is no other matrix B different than A in $\mathcal{M}$ such that $\mathsf{dat}(B) = \mathsf{dat}(A)$. Thus, if we sample a matrix M uniformly at

random from $\mathcal{M}$ there is a higher probability that $\mathsf{dat}(M) = \mathring{\mathcal{D}}$ than $\mathsf{dat}(M) = \mathsf{dat}(A)$, and our proof is complete. □

Lemma 3.2 says that, if one wants to sample datasets from a SLISP null model $\mathbf{\Pi} = (\mathcal{Z}, \pi)$, the samples obtained by using GMMT are actually drawn from a *distorted* null model $\tilde{\mathbf{\Pi}} = (\mathcal{Z}, \tilde{\pi})$, where $\tilde{\pi} \neq \pi$. As discussed in Sect. 2.1, these samples are intended to be used to compute key quantities (p-values and adjusted critical values) for evaluating the significance of results w.r.t. $\mathbf{\Pi}$, but given that they are not distributed according to π, any conclusion drawn using these samples could potentially be wrong. In the next section, we introduce DiFFuSR, our suite of algorithms to sample datasets from SLISP null models *without distortion*.

Before moving on, we mention that the root cause of the distortion is, informally, the choice to consider the order of the transactions in a dataset as meaningful for knowledge discovery tasks, when it is not: these tasks are effectively defined on datasets seen as *unordered* bags of transactions. Transaction identifiers are irrelevant for most knowledge discovery tasks. E.g., no pattern mining algorithm requires such identifiers. Thus, there is no reason to require the order of transactions to be fixed, and a null model that does not impose this constraint is more appropriate for mining results from transactional datasets than one that does.

3.2 Sampling Without Distortion

3.2.1 A First Algorithm

To warm up, we first present DiFFuSR-N (the N stands for "naïve"), a relatively simple algorithm to sample *without distortion* from a SLISP null model $\mathbf{\Pi} = (\mathcal{Z}, \pi)$ associated to an observed dataset $\mathring{\mathcal{D}}$. DiFFuSR-N uses MCMC with MH to sample from $\mathcal{M}$, the set of binary matrices also considered by GMMT (Sect. 3.1). The neighborhood structure over $\mathcal{M}$ is the same as in GMMT, as is the distribution η_M, $M \in \mathcal{M}$, which is the uniform distribution over the out-neighbors of M. The intuition behind DiFFuSR-N is that, if, for each dataset $\mathcal{D} \in \mathcal{Z}$, we can compute the number $\mathsf{c}(\mathcal{D})$ of matrices $M \in \mathcal{M}$ such that $\mathsf{dat}(M) = \mathcal{D}$, then we can remove the distortion that the presence of these "copies" introduces. The following result gives the expression for $\mathsf{c}(\mathcal{D})$.

Lemma 3.3. *For any dataset $\mathcal{D} \in \mathcal{Z}$, let $L_{\mathcal{D}} \doteq \{\ell_1, \ldots, \ell_{z_{\mathcal{D}}}\}$ be the set of the $z_{\mathcal{D}}$ distinct lengths of the transactions in $\mathcal{D}$. For each $1 \leq i \leq z_{\mathcal{D}}$, let T_i be the* bag *of transactions of length ℓ_i in $\mathcal{D}$. Let $\bar{T}_i = \{\tau_{i,1}, \ldots, \tau_{i,h_i}\}$ be the* set *of transactions of length ℓ_i in $\mathcal{D}$, i.e., without duplicates. For each $1 \leq j \leq h_i$, let $W_{i,j} \doteq \{t' \in T_i : t' = \tau_{i,j}\}$ be the* bag *of transactions equal to $\tau_{i,j}$ in T_i (including $\tau_{i,j}$). Then, the number of matrices M in $\mathcal{M}$ such that $\mathsf{dat}(M) = \mathcal{D}$ is*

$$\mathsf{c}(\mathcal{D}) \doteq \prod_{i=1}^{z_{\mathcal{D}}} \underbrace{\binom{|T_i|}{|W_{i,1}|, \ldots, |W_{i,h_i}|}}_{\textit{multinomial coefficient}} = \prod_{i=1}^{z_{\mathcal{D}}} \frac{|T_i|!}{\prod_{j=1}^{h_i} |W_{i,j}|!} \ . \tag{3.4}$$

Proof. Recall that $\mathcal{M}$ depends on the observed dataset $\mathring{\mathcal{D}}$ and on the arbitrary ordering of its transactions chosen by GMMT, as the ordering fixes the row-sums r_i, $1 \leq i \leq m$ of the matrices in $\mathcal{M}$. In other words, it fixes the row indices of rows corresponding to transactions of length ℓ_i, $1 \leq i \leq z_{\mathcal{D}}$, of $\mathcal{D}$. Thus, the number of different ways in which the transactions of $\mathcal{D}$ can be assigned to the rows of a matrix in $\mathcal{M}$ is the product, over the lengths in $L_{\mathcal{D}}$, of the number q_i of different ways in which the transactions in T_i can be assigned, i.e.,

$$\mathsf{c}(\mathcal{D}) = \prod_{i=1}^{z_{\mathcal{D}}} q_i \ .$$

Thus, we only have to argue that

$$q_i = \binom{|T_i|}{|W_{i,1}|, \ldots, |W_{i,h_i}|},$$

which is true because the multinomial coefficient $\binom{n}{k_1,\ldots,k_h}$ is the number of different permutations of a bag containing n objects such that k_1 objects are indistinguishable among themselves and of type 1, k_2 objects are indistinguishable among themselves and of type 2, and so on [Stanley, 2011, Eq. 1.22]. □

The core idea of DiFfuSR-N is that the probability $\pi(\mathcal{D})$ of sampling $\mathcal{D}$ should be "spread" among the $\mathsf{c}(\mathcal{D})$ matrices $M \in \mathcal{M}$ s.t. $\mathsf{dat}(M) = \mathcal{D}$. Thus, DiFfuSR-N uses MCMC with MH on the state space $\mathcal{M}$, but with the desired stationary distribution

$$\xi(M) \doteq \frac{\pi(\mathsf{dat}(M))}{\mathsf{c}(\mathsf{dat}(M))} \ . \tag{3.5}$$

As mentioned, the probability distribution over the out-neighbors of a matrix is the uniform as in GMMT,

so DiFFuSR-N can also reuse GMMT's procedure to sample an out-neighbor M'' of a matrix M' [Gionis et al., 2007, Alg. 2]. Computing $\mathsf{c}(\mathsf{dat}(M''))$ using $\mathsf{c}(\mathsf{dat}(M'))$ is relatively straightforward, knowing the swap that allows to obtain M'' from M', thus only the number of copies for the initial state X_0 corresponding to $\mathring{\mathcal{D}}$ has to be computed from scratch using the definition in (3.4). Using the notation from the statement of Lemma 3.3, given a transaction $t \in \mathsf{dat}(M')$, suppose $t \in T_i$ for $1 \leq i \leq z_{\mathsf{dat}(M')}$. Further suppose $t = \tau_{i,j} \in \bar{T}_i$, where $1 \leq j \leq h_i$. Let net be a dictionary that maps each different transaction $t \in \mathsf{dat}(M')$ to $|W_{i,j}|$, i.e., the size of the bag of transactions equal to t (including t). This data structure is easy to initialize and keep up to date. We can then obtain $\mathsf{c}(\mathsf{dat}(M'))$ from $\mathsf{c}(\mathsf{dat}(M''))$ as shown in Alg. 1, which leverages the fact that $\mathsf{c}(\mathsf{dat}(M')) = \mathsf{c}(\mathsf{dat}(M''))$ if $\mathsf{dat}(M'') = \mathsf{dat}(M')$ (line 2), and the definition of the multinomial to greatly simplify the computation (lines 3–5).

Algorithm 1: Computing $\mathsf{c}(\mathsf{dat}(M''))$ from $\mathsf{c}(\mathsf{dat}(M'))$

Input: Value $\mathsf{c}(\mathsf{dat}(M'))$, valid swap $((t_a, a), (t_b, b))$ from $\mathsf{dat}(M')$ to $\mathsf{dat}(M'')$, dictionary net
Output: $\mathsf{c}(\mathsf{dat}(M''))$

1 $\bar{t}_a \leftarrow (t_a \setminus \{a\}) \cup \{b\}$, $\bar{t}_b \leftarrow (t_b \setminus \{b\}) \cup \{a\}$
2 **if** $t_a = \bar{t}_b$ **then return** $\mathsf{c}(\mathsf{dat}(M'))$
3 **foreach** $i \in \{a, b\}$ **do**
4 **if** net *has key* $\bar{t}_i$ **then** $\beta_i \leftarrow \mathsf{net}[\bar{t}_i]$ **else** $\beta_i \leftarrow 0$
5 **return** $\mathsf{c}(\mathsf{dat}(M'))\frac{\mathsf{net}[t_a]\mathsf{net}[t_b]}{(\beta_a+1)(\beta_b+1)}$

DiFFuSR-N obtains a matrix M by running MCMC sampling with MH with the inputs just described, and outputs the dataset $\mathsf{dat}(M)$.

Theorem 3.4. *The output of DiFFuSR-N is a sample from $\mathcal{Z}$ distributed according to π.*

Proof. A dataset $\mathcal{D} \in \mathcal{Z}$ is returned in output by DiFFuSR-N iff a matrix M such that $\mathsf{dat}(M) = \mathcal{D}$ is sampled. There are $\mathsf{c}(\mathcal{D})$ such matrices M, each with a probability $\xi(M)$ as in (3.5) to be sampled, from the properties of MCMC sampling with MH. Thus, the probability of returning $\mathcal{D}$ is exactly $\pi(\mathcal{D})$. □

3.2.2 A Refined Algorithm

We now discuss a more refined algorithm DiFFuSR-R (the R is for "refined") to sample datasets from a SLISP null model $\Pi = (\mathcal{Z}, \pi)$. DiFFuSR-R is also based on MCMC sampling with MH, but uses a Markov chain with a different state space and neighborhood structure than GMMT and DiFFuSR-N.

DIFFUSR-R *directly* removes the distortion from the state space, rather than counteracting it at the level of the stationary distribution of the Markov chain like DIFFUSR-N does.

The state space of DIFFUSR-R is the dataset null set $\mathcal{Z}$ itself, without any representation of the datasets as matrices or imposition of ordering of the transactions in a dataset which is irrelevant for many knowledge discovery tasks (see end of Sect. 3.1).

In the initial neighborhood structure, there is an edge from a dataset $\mathcal{D}' \in \mathcal{Z}$ to another dataset $\mathcal{D}'' \in \mathcal{Z}$ if there exist two transactions $t_a, t_b \in \mathcal{D}'$ such that *(1)* neither $t_a \setminus t_b$ nor $t_b \setminus t_a$ are empty, and at least one of these two asymmetric set differences contains strictly more than one item; and *(2)* there are items $a \in t_a \setminus t_b$ and $b \in t_b \setminus t_a$ such that if we let $\bar{t}_a \doteq (t_a \setminus \{a\}) \cup \{b\}$ and $\bar{t}_b \doteq (t_b \setminus \{b\}) \cup \{a\}$, then it holds that $\mathcal{D}'' = (\mathcal{D}' \setminus \{t_a, t_b\}) \cup \{\bar{t}_a, \bar{t}_b\}$, i.e., $\mathcal{D}''$ can be obtained from $\mathcal{D}'$ by swapping a pair of items between two transactions that are not one a subset (proper or improper) of the other, and each of which contains only one of the two items. This operation can be seen as a *dataset-oriented* version of the *swap* at the basis of the swap randomization process used by GMMT, thus we still refer to it as *swap*. A swap can be identified as an unordered pair $((t_a, a), (t_b, b))$. The condition on one of the asymmetric set differences containing strictly more than one item ensures that a swap always "leads" to a different dataset, i.e., there are no swaps from $\mathcal{D}'$ to $\mathcal{D}'$. On the other hand, there may be multiple swaps that lead from $\mathcal{D}'$ to a different $\mathcal{D}''$. We denote the number of such swaps with $\mathsf{sw}_{\mathcal{D}'}(\mathcal{D}'')$ and show, in Lemma 3.5 and Corol. 3.5.1, expressions for this quantity and for $\mathsf{sw}_{\mathcal{D}''}(\mathcal{D}')$ respectively, as both are needed by DIFFUSR-R, as we explain later.

For any $t_1, t_2 \in \mathcal{D}'$, let $\chi(t_1, t_2) \doteq 2$ if $|t_1| = |t_2|$ and $|t_1 \setminus t_2| = |t_2 \setminus t_1| = 2$, and $\chi(t_1, t_2) \doteq 1$ otherwise.

Lemma 3.5. *Let $((t_a, a), (t_b, b))$ be a swap from $\mathcal{D}'$ to $\mathcal{D}''$. Then,*

$$\mathsf{sw}_{\mathcal{D}'}(\mathcal{D}'') = \chi(t_a, t_b) \left|\{t \in \mathcal{D}' : t = t_a\}\right| \cdot \left|\{t \in \mathcal{D}' : t = t_b\}\right| \ .$$

Proof. We start by showing that, independently from the value of $\chi(t_a, t_b)$, all the swaps from $\mathcal{D}'$ to $\mathcal{D}''$ must involve transactions that are identical to t_a and t_b, i.e, be in the form $((t_a, x), (t_b, y))$, for some $x, y \in \mathcal{I}$. Let $((t_c, c), (t_d, d))$ be a swap from $\mathcal{D}'$ to $\mathcal{D}''$. We want to show that $((t_c, c), (t_d, d)) = ((t_a, a), (t_b, b))$.

Let $\bar{t}_a = (t_a \setminus \{a\}) \cup \{b\}$, $\bar{t}_b = (t_b \setminus \{b\}) \cup \{a\}$, $\bar{t}_c = (t_c \setminus \{c\}) \cup \{d\}$, and $\bar{t}_d = (t_d \setminus \{d\}) \cup \{c\}$. Then,

$$(\mathcal{D}' \setminus \{t_a, t_b\}) \cup \{\bar{t}_a, \bar{t}_b\} = (\mathcal{D}' \setminus \{t_c, t_d\}) \cup \{\bar{t}_c, \bar{t}_d\}$$

since both sides are equal to $\mathcal{D}''$. Since datasets are *bags*, the operations of difference and union are effectively one the inverse of the other. Thus, we can write

$$\mathcal{D}' \cup \{\bar{t}_a, \bar{t}_b\} \cup \{t_c, t_d\} = \mathcal{D}' \cup \{\bar{t}_c, \bar{t}_d\} \cup \{t_a, t_b\} \ .$$

We can remove $\mathcal{D}'$ from both sides to get

$$\{\bar{t}_a, \bar{t}_b, t_c, t_d\} = \{\bar{t}_c, \bar{t}_d, t_a, t_b\} \ . \tag{3.6}$$

By definition of a swap, it must be $t_a \neq \bar{t}_a$ and $t_a \neq \bar{t}_b$, thus it must be either $t_a = t_c$ or $t_a = t_d$, and similarly for t_b.

Assume now that $|t_a| \neq |t_b|$, then only one between $t_a = t_c$ and $t_a = t_d$ is actually possible so, w.l.o.g., let $t_a = t_c$ and $t_b = t_d$. Then from these facts and (3.6), it must be $\{\bar{t}_a, \bar{t}_b\} = \{\bar{t}_c, \bar{t}_d\}$, thus either $\bar{t}_a = \bar{t}_c$ or $\bar{t}_a = \bar{t}_d$. Since $|\bar{t}_a| = |t_a| = |t_c| = |\bar{t}_c|$ and $|\bar{t}_d| = |t_d| = |t_b|$, then it can only be $\bar{t}_a = \bar{t}_c$, which implies $c = a$, and similarly it must be $b = d$. Thus, if $|t_a| \neq |t_b|$, then all swaps from $\mathcal{D}'$ to $\mathcal{D}''$ are equal to $((t_a, a), (t_b, b))$, which implies the thesis for this case.

Assume now that $|t_a| = |t_b|$ but $|t_a \setminus t_b|$ and $|t_b \setminus t_a|$ are different than 2 (the two asymmetric set differences must have the same size, since the two sets have the same size). From the first part of the proof, we have that it must be either $t_a = t_c$ or $t_a = t_d$, and similarly for t_b. W.l.o.g., let $t_a = t_c$ and $t_b = t_d$. If $|t_a \setminus t_b| = |t_b \setminus t_a| = 1$, then a and b are the only two items that can be swapped, so it must be $c = a$ and $b = d$. Thus, in this specific subcase, all swaps from $\mathcal{D}'$ to $\mathcal{D}''$ are equal to $((t_a, a), (t_b, b))$, which implies the thesis for this case. Assume now that $|t_a \setminus t_b| = |t_b \setminus t_a| > 2$. From these facts and (3.6), it must be $\{\bar{t}_a, \bar{t}_b\} = \{\bar{t}_c, \bar{t}_d\}$, thus either $\bar{t}_a = \bar{t}_c$ or $\bar{t}_a = \bar{t}_d$. We now show that it must be $\bar{t}_a = \bar{t}_c$. Assume by

contradiction that $\bar{t}_a = \bar{t}_d$. Since $t_c = t_a$, it holds $\bar{t}_a = t_c \setminus \{a\} \cup \{b\}$. Thus, it must be

$$(t_c \setminus \{a\}) \cup \{b\} = (t_d \setminus \{d\}) \cup \{c\} \ .$$

Since $a, c \in t_c$ and $b \notin t_c$, by taking the asymmetric set differences between each side of the above identity and t_c we get

$$\{b\} = (t_d \setminus \{d\}) \setminus t_c$$

which means that $t_d \setminus t_c = t_b \setminus t_a = \{b, d\}$, i.e., this set difference has size equal to 2, which is a contradiction since we assumed that $|t_b \setminus t_a| > 2$. Thus, it must be $\bar{t}_a = \bar{t}_c$, which implies $c = a$, and similarly it must be $b = d$. Thus, all swaps from $\mathcal{D}'$ to $\mathcal{D}''$ are equal to $((t_a, a), (t_b, b))$, which implies the thesis for this case.

We have therefore shown that the thesis is true for the case $\chi(t_a, t_b) = 1$. Consider now the other case, which means that $|t_a| = |t_b|$ and $|t_a \setminus t_b| = |t_b \setminus t_a| = 2$. W.l.o.g., let $t_a \setminus t_b = \{a, z\}$ and $t_b \setminus t_a = \{b, w\}$. It is easy to see that the swaps $((t_a, a), (t_b, b))$ and $((t_a, z), (t_b, w))$ lead to the same dataset $\mathcal{D}''$, while the other two swaps $((t_a, a), (t_b, w))$ $((t_a, z), (t_b, b))$ lead to a different dataset. Thus, there are $\chi(t_a, t_b) = 2$ swaps from $\mathcal{D}'$ to $\mathcal{D}''$ involving the transactions t_a and t_b. From the first part of the proof, we have that all swaps from $\mathcal{D}'$ to $\mathcal{D}''$ must involve transactions identical to t_a and t_b, hence we obtain the thesis for this case. □

Corollary 3.5.1. *Let $((t_a, a), (t_b, b))$ be a swap from $\mathcal{D}'$ to $\mathcal{D}''$, and $\bar{t}_a \doteq (t_a \setminus \{a\}) \cup \{b\}$ and $\bar{t}_b \doteq (t_b \setminus \{b\}) \cup \{a\}$. Then, $\mathsf{sw}_{\mathcal{D}''}(\mathcal{D}') = \chi(t_a, t_b)(|\{t \in \mathcal{D}' : t = \bar{t}_a\}| + 1)(|\{t \in \mathcal{D}' : t = \bar{t}_b\}| + 1)$.*

We need the following result on the number of valid swaps from a dataset $\mathcal{D}$.

Theorem 3.6. *Let $M_{\mathcal{D}}$ be any matrix representation of $\mathcal{D}$ (Sect. 3.1). Then,*

$$\mathsf{dts}(\mathcal{D}) = \underbrace{\mathsf{mts}(M_{\mathcal{D}})}_{\textit{Thm. 3.1}} - \mathsf{H}(\mathcal{D}), \tag{3.7}$$

where

$$\mathsf{H}(\mathcal{D}) \doteq \frac{1}{2}|\{(t_a, t_b) \in \mathcal{D} \times \mathcal{D} : |t_a| = |t_b| \wedge |t_a \setminus t_b| = |t_b \setminus t_a| = 1\}| \ . \tag{3.8}$$

Proof. Let $((t_a, a), (t_b, b))$ be any of the $\mathsf{mts}(M_{\mathcal{D}})$ swaps from $M_{\mathcal{D}}$ to any M of its neighbor matrices in

the Markov chain used by GMMT. From the definitions of "swap" for GMMT and for DiFfuSR-R, it follows that $((t_a, a), (t_b, b))$ is a valid swap for DiFfuSR-R iff $\mathsf{dat}(M) \neq \mathcal{D}$, i.e., if the dataset corresponding to M is not $\mathcal{D}$ itself. The number of valid swaps from $\mathcal{D}$ is therefore $\mathsf{mts}(M_{\mathcal{D}})$ minus the total number of swaps from $M_{\mathcal{D}}$ to any M of its neighbors with $\mathsf{dat}(M) = \mathcal{D}$. We now show that the number of such "self-swaps" is $\mathsf{H}(\mathcal{D})$.

For a neighbor $M \neq M_{\mathcal{D}}$ of $M_{\mathcal{D}}$ to have $\mathsf{dat}(M) = \mathcal{D}$, it must be the case that there is a swap $((t_a, a), (t_b, b))$ such that

$$(t_a \setminus \{a\}) \cup \{b\} = t_b \text{ and } (t_b \setminus \{b\}) \cup \{a\} = t_a \ .$$

Since $a \in t_a \setminus t_b$ and $b \in t_b \setminus t_a$, the above is possible iff $|t_a| = |t_b|$ and $\{a\} = t_a \setminus t_b$ and $\{b\} = t_b \setminus t_a$. The number of such *unordered* pairs of transactions (i.e., the number of self-swaps) is as in (3.8), where the factor $1/2$ is to correct for the fact that the considered set contains *ordered* pairs of transactions. □

DiFfuSR-R uses MCMC sampling with MH to sample according to π. Let $\mathcal{D}' \in \mathcal{Z}$ be the current state of the chain. The distribution $\eta_{\mathcal{D}'}$, according to which MH samples a neighbor $\mathcal{D}''$ of $\mathcal{D}'$ is

$$\eta_{\mathcal{D}'}(\mathcal{D}'') \doteq \frac{\mathsf{sw}_{\mathcal{D}'}(\mathcal{D}'')}{\mathsf{dts}(\mathcal{D}')} \ . \tag{3.9}$$

Sampling a neighbor of $\mathcal{D}'$ according to the distribution $\eta_{\mathcal{D}'}$ is straightforward (see Alg. 2). Let $\mathcal{G}$ be the bag of ordered pairs (t, i) for every $t \in \mathcal{D}'$ and $i \in t$. We can draw two pairs (t_a, a) and (t_b, b) independently and uniformly at random from $\mathcal{G}$ and check whether the unordered pair $((t_a, a), (t_b, b))$ is a valid swap, i.e., it satisfies all the requirements for t_a, t_b, a, and b in the definition of a swap (there are pairs that are not valid swaps, e.g., those involving the same transaction). In this case the sampled neighbor $\mathcal{D}''$ is the one obtained by performing the swap, otherwise two more elements are drawn from $\mathcal{G}$ until a valid swap is found.

Lemma 3.7. *Let $\mathcal{D}''$ be a neighbor of $\mathcal{D}' \in \mathcal{Z}$. The probability that the above procedure outputs $\mathcal{D}''$ is $\eta_{\mathcal{D}'}(\mathcal{D}'')$.*

Proof. The procedure keeps sampling candidate swaps (i.e., pairs of elements of $\mathcal{G}$) until it finds one that

is a valid swap. Since each element of $\mathcal{G}$ is sampled independently and uniformly at random from $\mathcal{G}$, then the candidate swap is chosen uniformly at random among all candidate swaps, thus, if it is valid, is also chosen uniformly at random among all valid swaps, i.e., each valid swap has a probability of $1/\mathsf{dts}(\mathcal{D}')$ of being chosen among all valid swaps, as there is one and only one pair of elements of $\mathcal{G}$ for each valid swap. $\mathcal{D}''$ is output whenever the procedure chooses one of the $\mathsf{sw}_{\mathcal{D}'}(\mathcal{D}'')$ valid swaps leading to it from $\mathcal{D}'$. Thus, the probability of outputting $\mathcal{D}''$ is the sum of the probabilities of choosing each of these swaps, hence the thesis. □

Assuming the current state of the Markov chain is $\mathcal{D}'$, by plugging the definition of $\eta_{\mathcal{D}'}$ from (3.9) into (2.2) (using $\xi = \pi$), we get that the probability MH accepts as the next state a neighbor $\mathcal{D}''$ drawn according to $\eta_{\mathcal{D}'}$ is

$$\min\left\{1, \frac{\pi(\mathcal{D}'')\mathsf{sw}_{\mathcal{D}''}(\mathcal{D}')\mathsf{dts}(\mathcal{D}')}{\pi(\mathcal{D}')\mathsf{sw}_{\mathcal{D}'}(\mathcal{D}'')\mathsf{dts}(\mathcal{D}'')}\right\} \quad . \tag{3.10}$$

DiFfuSR-R needs data structures and procedures to compute all the necessary quantities. More precisely, DiFfuSR-R initializes some data structures at the beginning of its execution, and uses and updates them to select the successive states of the Markov chain (which starts at $\mathring{\mathcal{D}}$). First, the bag $\mathcal{G} \doteq \{(t, i) : t \in \mathcal{D}', i \in t\}$ is built in the obvious way, with $\mathcal{D}' = \mathring{\mathcal{D}}$. This bag is all that is needed to sample a neighbor from the current state $\mathcal{D}'$ according to the distribution $\eta_{\mathcal{D}'}$. Updating $\mathcal{G}$ after moving to a new state $\mathcal{D}''$ is straightforward: only pairs involving the transactions participating in the performed swap are removed (for the transactions before the swap) or added (for the transactions after the swap). Secondly, DiFfuSR-R builds a dictionary M s.t. for each transaction $t \in \mathring{\mathcal{D}}$,

$$\mathsf{M}[t] \doteq \left|\{t' \in \mathring{\mathcal{D}} : t' = t\}\right|$$

Algorithm 2: Sampling according to the refined neighborhood distribution $\eta_{\mathcal{D}'}(\mathcal{D}'')$

Input: bag $\mathcal{G} \doteq \{(t, i) : t \in \mathcal{D}, i \in t\}$
Output: valid swap $((t_a, a), (t_b, b))$ from $\mathcal{D}'$ to $\mathcal{D}''$

do
  $(t_a, a), (t_b, b) \leftarrow$ uniform independent samples from $\mathcal{G}$
while $(a \in t_b) \vee (b \in t_a) \vee (t_a \setminus \{a\} = t_b \setminus \{b\})$
return $((t_a, a), (t_b, b))$

is the number of transactions in $\mathring{\mathcal{D}}$ equal to t, including t. Computing $\mathsf{sw}_{\mathcal{D}'}(\mathcal{D}'')$ and $\mathsf{sw}_{\mathcal{D}''}(\mathcal{D}')$ for any dataset $\mathcal{D}'$ and any $\mathcal{D}''$ of its neighbors is straightforward using the expressions of these quantities from Lemma 3.5 and Corol. 3.5.1 and the dictionary M. Updating M after moving to a new state is easy, requiring changes only to the key-value pairs where the key is a transaction involved in the swap, either before or after the swap. The last quantities to discuss are $\mathsf{mts}(M_{\mathcal{D}'})$ and $\mathsf{H}(\mathcal{D}')$ from (3.7). Gionis et al. [2007, Thm. 4.3, Corol. 4.4] give procedures to compute $\mathsf{mts}(M_{\mathcal{D}'})$ and, using this quantity, $\mathsf{mts}(M_{\mathcal{D}''})$, thus we can focus on $\mathsf{H}(\mathcal{D}')$ and how to obtain $\mathsf{H}(\mathcal{D}'')$ from it. Let TL be a dictionary that maps each different $\ell \in \mathbb{N}$ for which there is at least one transaction $t \in \mathring{\mathcal{D}}$ with $|t| = \ell$ to the *bag* of transactions in $\mathring{\mathcal{D}}$ with length ℓ. Initializing this data structure given $\mathring{\mathcal{D}}$ is straightforward, and at the same time one can also compute $\mathsf{H}(\mathring{\mathcal{D}})$. Given TL and $\mathsf{H}(\mathcal{D}')$ for the current state $\mathcal{D}'$, we can compute $\mathsf{H}(\mathcal{D}'')$ for a neighbor $\mathcal{D}''$ obtained by performing a swap $((t_a, a), (t_b, b))$ from $\mathcal{D}'$ as follows (pseudocode in Alg. 3). First, a variable z is initialized to $\mathsf{H}(\mathcal{D}')$ (line 1). Then, for each $t \in \{t_a, t_b\}$ we iterate over $\mathsf{TL}[|t|] \setminus \{t_a, t_b\}$, i.e., the bag of transactions of length $|t|$ that are neither t_a nor t_b. Let t^* be a transaction in this bag, and let d^* be $|(t \setminus t^*) \cup (t^* \setminus t)|$, i.e., the number of items that are either only in t or only in t^* (line 4). Then, the algorithm checks if d^* is not larger than four (line 5): when $d^* > 4$, then there is no self-swap involving t and t^* from $\mathcal{D}'$ to itself, nor there is one from $\mathcal{D}''$ to itself, thus there is nothing to do in this case. If instead $d^* \leq 4$, the procedure computes the number of items that are either only in t^* or in the transaction $\bar{t}$ obtained from t after the swap (i.e., if $t = t_a$, then $\bar{t} = (t_a \setminus \{a\}) \cup \{b\}$, and similarly if $t = t_b$): for each $i \in \{a, b\}$ (line 7), if i appears in both t and t^* or does not appear in either, then d is increased by one (line 8), as that means that i will appear in exactly one of $\bar{t}$ and t^* after the swap, otherwise d is decreased by one (line 9), because i will either appear or not appear in both $\bar{t}$ and t^* after the swap. Then, if d equals 2 (resp. is not equal to 2) and d^* is not equal to 2 (resp. equals 2), it means that there is an additional (resp. one fewer) invalid self-swap that would lead from $\mathcal{D}''$ to itself, and the variable z is updated accordingly (lines 10–11).

3.3 Extensions and Discussion

In Ch. 4, we use DiFFuSR to test the significance of the number $|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|$ of frequent itemsets (as described in Sect. 2.1). This is a relatively simple task that we choose to make the presentation self-contained and

Algorithm 3: Computing $\mathsf{H}(\mathcal{D}'')$ from $\mathsf{H}(\mathcal{D}')$

Input: Value $\mathsf{H}(\mathcal{D}')$, valid swap $((t_a, a), (t_b, b))$ from $\mathcal{D}'$ to $\mathcal{D}''$, dictionary TL
Output: $\mathsf{H}(\mathcal{D}'')$

```
z ← H(D')
foreach t ∈ {t_a, t_b} do
    foreach t* ∈ TL[|t|] \ {t_a, t_b} do
        d* ← |(t \ t*) ∪ (t* \ t)|
        if d* ≤ 4 then
            d ← d*
            foreach i ∈ {a, b} do
                if i ∈ t ∩ t* ∨ i ∉ t ∪ t* then d ← d + 1
                else d ← d − 1
            if d = 2 ∧ d* ≠ 2 then z ← z + 1
            else if d ≠ 2 ∧ d* = 2 then z ← z − 1
return z
```

accessible: our algorithms can be used for assessing the statistical significance of the results of many other tasks defined on transactional datasets, including significant pattern mining [Pellegrina et al., 2019b]. Hämäläinen and Webb [2019] discuss many such tasks, as do Gionis et al. [2007], who show how to use GMMT for them. DiFFuSR can be used as a drop-in replacement of GMMT in these cases.

Algorithms for sampling different kinds of datasets for knowledge discovery tasks also incur a distortion of the sample space similar to the one we discussed for GMMT. For example, Tonon and Vandin [2019] present swap-randomization methods to draw *sequential* datasets (approximately) uniformly at random from a model, which are then used for mining significant frequent sequences. Their sampling methods, being similar to GMMT, suffer from a similar distortion. It is straightforward to develop variants of DiFFuSR for sampling sequential datasets and warrants future work. Similar extensions should also be possible for other kinds of patterns.

A transactional dataset $\mathcal{D}$ can be seen as a bipartite graph $G_{\mathcal{D}}$ with one vertex for each item $i \in \mathcal{I}$ on one side, and one vertex for each transaction $t \in \mathcal{D}$ on the other side, and such that there is an edge (i, t) iff $i \in t$ [Gionis et al., 2007, Sect. 4.1]. GMMT can be seen as an algorithm for sampling, uniformly, bipartite graphs with the same degree distribution as $G_{\mathcal{D}}$. In this setting, the distortion means that two bipartite graphs that are identical up to permutations of the identifiers of the transaction vertices (a weak

form of graph isomorphism) would be considered different graphs, which may or may not be desired by the user, who is the ultimate decision maker and should be made aware of the consequences of choosing different null models. DiFFuSR can be used to fix this distortion.

Is distortion always undesirable? If the user's goal is to sample from a null model where different orderings of the transactions correspond to different datasets, then there is no distortion in this case and the question is moot. However, distortion is undesirable if the goal is to sample from a SLISP null model. Distortion is *always* present and undesirable when the user samples according to a null model distribution π from a null set X, but their true objective is to sample according to π from a null set Y, and there does *not* exist an bijective function between X and Y.

One limitation of this work is that we do not show an upper bound to the *mixing time* $\mathsf{t}(\varepsilon)$, $\varepsilon \in (0, 1)$ of the Markov chains of our algorithms, i.e., the number of steps needed for the distribution of the state X_t to have total variation distance at most ε from the desired stationary distribution π [Mitzenmacher and Upfal, 2005, Ch. 10]. Such a bound is not available for GMMT either, because using the MH approach makes such a derivation particularly challenging. We conjecture that better bounds to $\mathsf{t}(\varepsilon)$ can be derived for DiFFuSR-R than for DiFFuSR-N, thanks to its smaller and more connected state space.

Another limitation, in common with all previous work, is in the definition of the SLISP null model: while preserving the size, transaction lengths, and item supports is crucial, it may not be sufficient for the null set $\mathcal{Z}$ to be representative of the unknown process. Defining more descriptive null models and efficient procedures to sample from them is a interesting research direction.

Chapter 4

Experimental Evaluation

Our experimental evaluation focuses on three aspects. First, verify the presence of the *distortion* in the dataset null space $\mathcal{Z}$ by looking at the distribution of $\mathsf{c}(\mathsf{dat}(M))$ across the matrices state space $\mathcal{M}$, and assess the impact of such distortion on testing data mining results. Second, evaluate the *speed and scalability* of DiFfuSR by measuring the *step time* of DiFfuSR, i.e., the time to take a step on the Markov chain, and how it changes as the number $|\mathring{\mathcal{D}}|$ of transactions in the datasets grows. Third, empirically estimate the *mixing time* of DiFfuSR, i.e., the number of swaps for the distribution of the chain state to be close to the stationary distribution.

Table 4.1 Dataset statistics: number of transactions $|\mathring{\mathcal{D}}|$, number of items $|\mathcal{I}|$, density $\text{avg}|t|/|\mathcal{I}|$, where avg$|t|$ is the average transaction length, sum $w \doteq \sum_{i=1}^{m} |t_i|$ of transaction lengths, support threshold θ used in some experiments, and number of frequent itemsets w.r.t. θ.

Dataset $\mathring{\mathcal{D}}$	$\lvert\mathring{\mathcal{D}}\rvert$	$\lvert\mathcal{I}\rvert$	$\frac{\text{avg}\lvert t\rvert}{\lvert\mathcal{I}\rvert}$	w	θ	$\lvert\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)\rvert$
Foodmart	4,141	1,559	0.0028	18,319	2	4,247
Chess	3,196	75	0.4933	118,252	2,557	8,227
Mushroom	8,416	119	0.1933	193,568	2,525	2,587
BMS 1	59,602	497	0.0051	149,639	60	3,991
BMS 2	77,512	3,340	0.0014	358,278	156	3,683

4.1 Implementation, Environment, and Datasets

We implement DiFfuSR and GMMT in Java 8. All our code, together with instructions and a script to reproduce all our results and figures, is available from `https://github.com/acdmammoths/DiFfuSR`.[1] We run our experiments on an x86–64 AWS EC2 instance with the Amazon Linux 2 OS, 128GB of RAM, and 32 vCPUs. We use five publicly available[2] datasets, whose relevant statistics are in Table 4.1 and descriptions are detailed below.

- Foodmart: customer transactions from a retail store, obtained and transformed from SQL-Server 2000.
- Chess: a conversion of the UCI chess dataset.
- Mushroom: a conversion of the UCI mushroom dataset.
- BMS WebView 1 (BMS 1): click-stream data from a webstore used in KDD-Cup 2000, which has been prepared for itemset mining.
- BMS WebView 2 (BMS 2): click-stream data from a webstore used in KDD-Cup 2000, which has been prepared for itemset mining.

Table 4.2 Distortion: minimum, 1st quartile, median, 3rd quartile, and maximum of $\lfloor \ln(\mathsf{c}(\mathsf{dat}(M))) \rfloor$ across 10,000 states $M \in \mathcal{M}$.

	Distribution of $\lfloor \ln(\mathsf{c}(\mathsf{dat}(M))) \rfloor$				
Dataset	min.	Q1	med.	Q3	max.
Foodmart	21,848	21,851	21,852	21,855	21,861
Chess	22,589	22,596	22,598	22,598	22,599
Mushroom	67,449	67,580	67,628	67,639	67,649
BMS 1	343,570	345,695	347,551	349,260	350,721
BMS 2	541,598	542,301	542,926	543,515	544,058

[1] Please ask Prof. Matteo Riondato for access.
[2] `https://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php`

4.2 Distortion

As discussed in Sect. 3.1, the distortion in which GMMT incurs is due to the fact that not every dataset $\mathcal{D}$ has the same number $\mathsf{c}(\mathcal{D})$ of matrices $M \in \mathcal{M}$ such that $\mathcal{D} = \mathsf{dat}(M)$, or equivalently, that $\mathsf{c}(\mathsf{dat}(M))$ is not the same for every $M \in \mathcal{M}$. We verify the presence of this distortion by performing a (non-covering) random walk over $\mathcal{M}$ and tracking the value $\ln(\mathsf{c}(\mathsf{dat}(M)))$ for each visited state M. While a random walk may visit a state more than once, it never happened in our experiments. Additionally, the random walk bias towards higher-degree states has no influence on verifying the presence of distortion. In Table 4.2 we report quartiles of the observed distribution over 10,000 steps, without the decimal part due to lack of space. The results show the clear presence of distortion, given the large variability in $\mathsf{c}(\mathsf{dat}(M))$.

4.3 Impact of Distortion

DiFFuSR is a drop-in replacement for GMMT to validate data mining results as described by Hämäläinen and Webb [2019], Gionis et al. [2007], but the two approaches sample from two different null models: SLISP *with* distortion (GMMT), and *without* (DiFFuSR). Thus, it is not surprising that the outcome of the results validation may differ, depending on which method is used, and therefore which null model is assumed. We tested the significance of the number $|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|$ of frequent patterns (see Sect. 2.1), using 2,048 sample datasets for each method to approximate the null distribution of this statistic. Not surprisingly (as discussed above), using GMMT or DiFFuSR results different empirical null distributions, which could lead to different p-values and hence to different outcomes of the test depending on the critical value. In our preliminary analysis using DiFFuSR for significant frequent itemset mining [Hämäläinen and Webb, 2019, Pellegrina et al., 2019b], we observed that the set of significant patterns found by GMMT in the Chess dataset with FWER $\delta = 0.05$, was extremely different (Jaccard index: 0.12) from the sets of significant patterns found by the two DiFFuSR methods. Once more, this result is evidence that the user must be extremely cautious in choosing the method and therefore the assumed null model: the meaning of significance depends on the null model, and comparing results obtained under different null models (e.g., to compare the statistical power of two procedures) has no meaning.

Table 4.3 Step time (in ms): minimum, 1st quartile, median, 3rd quartile, and maximum over 10,000 steps.

		step time (ms)				
Dataset	algorithm	min.	Q1	med.	Q3	max.
Foodmart	DiFfuSR-N	< 1	1	2	2	16
	DiFfuSR-R	1	2	2	2	32
	GMMT	1	1	2	2	24
Chess	DiFfuSR-N	4	5	5	5	24
	DiFfuSR-R	11	13	14	14	49
	GMMT	3	5	5	5	16
Mushroom	DiFfuSR-N	8	12	13	13	56
	DiFfuSR-R	22	28	30	33	82
	GMMT	7	9	9	10	47
BMS 1	DiFfuSR-N	19	25	27	29	63
	DiFfuSR-R	27	31	32	38	73
	GMMT	19	24	25	27	63
BMS 2	DiFfuSR-N	33	44	47	50	98
	DiFfuSR-R	50	55	56	57	103
	GMMT	38	47	49	51	97

4.4 Step Time

The *step time* is the time needed to obtain a valid swap, compute the MH acceptance probability, and transition to the next state if it is accepted. In Table 4.3 we report the distribution, over 10,000 steps, of this quantity. We show the results for GMMT *only* for comparison purposes: due to the distortion it incurs, GMMT is not to be preferred just because it appears faster.

The distribution for DiFfuSR-N is comparable to that of GMMT, while DiFfuSR-R is slightly slower. This is expected since the execution of GMMT and DiFfuSR-N are very similar, where the only additional work for DiFfuSR-N is to compute $\mathsf{c}(\mathsf{dat}(M''))$ from $\mathsf{c}(\mathsf{dat}(M'))$. DiFfuSR-R is slower because it needs to compute $\mathsf{H}(\mathcal{D}'')$ from $\mathsf{H}(\mathcal{D}')$, which takes time linear in the numbers of transactions with the same lengths as the ones involved in the swaps (line 3 of Alg. 3). For this same reason, DiFfuSR-R is relatively slower than the others on datasets where every transaction has the same length (Chess, Mushroom).

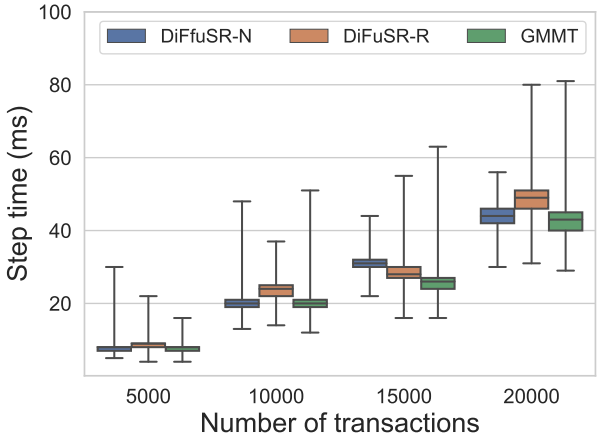


Figure 4.1 Scalability results: The step time distribution (in milliseconds) over 10,000 swaps for increasing values of $|\mathring{\mathcal{D}}|$. The line in each box corresponds to the median, the bottom and top of each box correspond to the first and third quartiles, and the lower and upper whiskers correspond to the minimum and maximum.

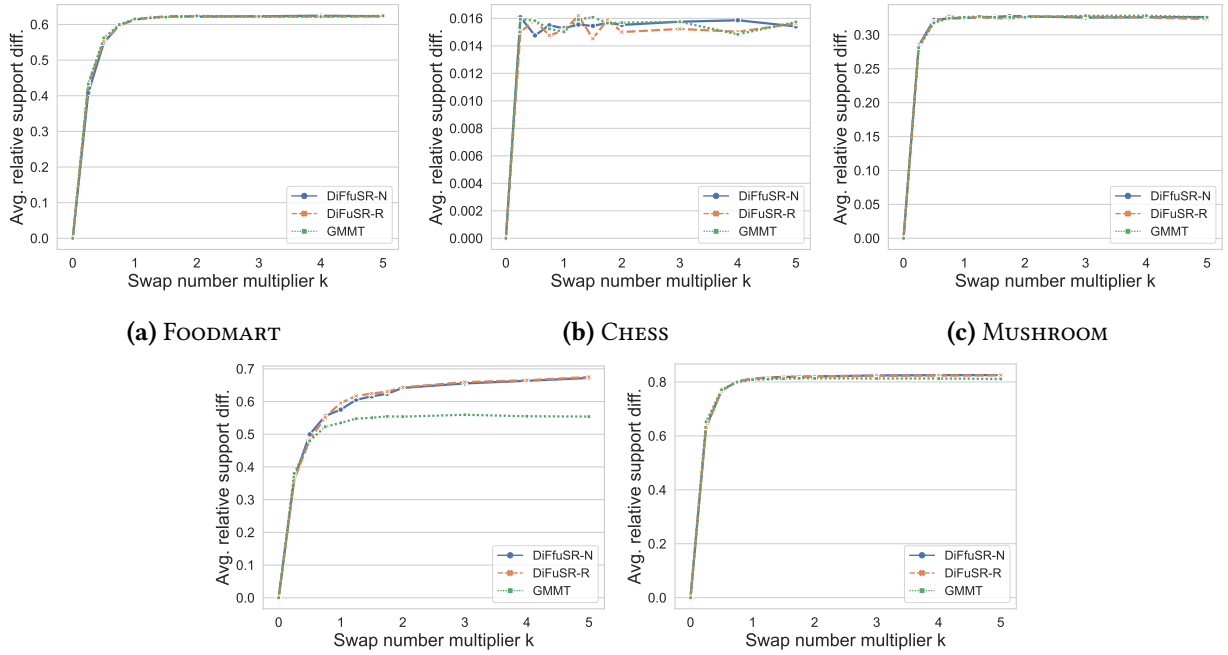
4.5 Scalability

We use the IBM Quest generator to create synthetic datasets with $|\mathring{\mathcal{D}}| \in \{5{,}000,\ 10{,}000,\ 15{,}000,\ 20{,}000\}$, on $|\mathcal{I}| = 100$ and average transaction length $|t| = 25$. We run all algorithms for 10,000 swaps on each dataset, and report the results in Fig. 4.1.

There is a linear relationship between the distribution of step times and the number of transactions, as all algorithms need to compute $\mathsf{mts}(M_{\mathcal{D}''})$, which takes time linear in $|\mathcal{D}''|$. The interquartile range ($Q3 - Q1$) grows in absolute terms because the individual step times grow, but it is essentially constant in relative terms.

4.6 Convergence to the Stationary Distribution

Since we cannot prove an upper bound to the mixing time of the Markov chains used by our algorithms (see Sect. 3.3), we empirically estimate it. Following other works using MCMC with MH for swap randomization [Tonon and Vandin, 2019], we estimate the mixing time by tracking the *Average Relative Support Difference (ARSD)*, defined as follows. Given the observed dataset $\mathring{\mathcal{D}}$, let $\theta \in (0, |\mathring{\mathcal{D}}|]$ be a minimum support threshold, and $\mathcal{D}_s$ be the dataset corresponding to the state of the chain after $s \in \mathbb{N}$ swaps. Then, the



(a) FOODMART **(b)** CHESS **(c)** MUSHROOM

(d) BMS 1 **(e)** BMS 2

Figure 4.2 Convergence results: $ARSD(\mathring{\mathcal{D}})$ as the swap number multiplier k grows, where k is s.t. the number of swaps is $s \doteq \lfloor k \sum_{i=1}^{m} |t_i| \rfloor$.

average relative support difference $ARSD(\mathcal{D}_s)$ of $\mathcal{D}_s$ is

$$ARSD(\mathcal{D}_s) \doteq \frac{1}{|\mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)|} \sum_{A \in \mathsf{FI}_{\mathring{\mathcal{D}}}(\theta)} \frac{|\sigma_A(\mathring{\mathcal{D}}) - \sigma_A(\mathcal{D}_s)|}{\sigma_A(\mathring{\mathcal{D}})} \ .$$

For each dataset, we report $ARSD(\mathcal{D}_s)$ for $s \doteq \lfloor kw \rfloor$ swaps, where $k \in \{0, 0.25, 0.50, \ldots, 2, 3, 4, 5\}$ and $w \doteq \sum_{i=1}^{m} |t_i|$, in Fig. 4.2. We use the values of θ from Table 4.1: the qualitative results do not change with other values.

We remark that comparing the mixing times of Markov chains with different stationary distributions (as GMMT and DIFFUSR-N) and even on different sets of states (as DIFFUSR-R vs. the other two) is *meaningless*, as they allow to sample different objects from different sets according to different distributions. Neither are the values of the ARSD comparable, as they are *proxies* for the mixing times, but not for the distance between the state distribution and the stationary distribution, and they also depend on the chosen θ. Therefore, we do not make such comparisons and only include the results from GMMT for completeness

(the mixing time for GMMT is the same observed by Gionis et al. [2007, Sect. 5.1]).

Figure 4.2 shows that in all cases, the ARSD stabilizes by $s = 2w$ swaps or earlier (by $s = w$), i.e., the mixing time appears to be linear in w. For CHESS, the fluctuations in the ARSD may seem large due to the scale of the y-axis, which is much smaller in Fig. 4.2b than in the other subfigures.

Chapter 5

Conclusion

We present DiFfuSR, a collection of algorithms for sampling unlabeled transactional datasets from a null model, which is a key step in evaluating the statistical validity of the results of knowledge discovery tasks. DiFfuSR uses swap-randomization, a Markov-Chain-Monte-Carlo approach. In contrast with previous work, DiFfuSR does not suffer from a distortion of the space of datasets, which was due to an assumed 1-to-1 mapping of the datasets to binary matrices. This distortion may (and does) affect the outcome of the validation, i.e., results in false discoveries. We avoid this assumption and show two methods (DiFfuSR-N and DiFfuSR-R) for drawing datasets from the non-distorted null space. Our experimental evaluation shows that the distortion is relevant, but DiFfuSR does not suffer from it, is fast, and scales well as the dataset grows.

Interesting directions for future work include the definition of richer null models that capture more important properties of the data, and fast algorithms to sample from these null models.

Bibliography

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th Int. Conf. Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8.

Carlo Emilio Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del Regio Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, SIGMOD '97, pages 265–276, 1997.

Wouter Duivesteijn and Arno Knobbe. Exploiting false discoveries–statistical validation of patterns and quality measures in subgroup discovery. In *2011 IEEE 11th International Conference on Data Mining*, pages 151–160. IEEE, 2011.

Egil Ferkingstad, Lars Holden, and Geir Kjetil Sandve. Monte Carlo null models for genomic data. *Statistical Science*, 30(1):59–71, 2015. ISSN 08834237, 21688745. URL `http://www.jstor.org/stable/24780405`.

Aristides Gionis, Heikki Mannila, Taneli Mielikäinen, and Panayiotis Tsaparas. Assessing data mining results via swap randomization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(3):14, 2007.

Stephan Günnemann, Phuong Dao, Mohsen Jamali, and Martin Ester. Assessing the significance of data mining results on graphs with feature vectors. In *2012 IEEE 12th International Conference on Data Mining*, pages 270–279, 2012. doi: 10.1109/ICDM.2012.70.

Wilhelmiina Hämäläinen. StatApriori: an efficient algorithm for searching statistically significant association rules. *Knowledge and Information Systems*, 23(3):373–399, 2010. doi: 10.1007/s10115-009-0229-8. URL https://doi.org/10.1007/s10115-009-0229-8.

Wilhelmiina Hämäläinen and Geoffrey I. Webb. A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery*, 33(2):325–377, 2019.

Sami Hanhijärvi. Multiple hypothesis testing in pattern discovery. In *International Conference on Discovery Science*, pages 122–134. Springer, 2011.

Sami Hanhijärvi, Gemma C. Garriga, and Kai Puolamäki. Randomization techniques for graphs. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, SDM '09, pages 780–791, 2009. doi: 10.1137/1.9781611972795.67. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972795.67.

Sami Hanhijärvi, Markus Ojala, Niko Vuokko, Kai Puolamäki, Nikolaj Tatti, and Heikki Mannila. Tell me something I don't know: Randomization strategies for iterative data mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 379–388, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557065. URL https://doi.org/10.1145/1557019.1557065.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6 (2):65–70, 1979.

Adam Kirsch, Michael Mitzenmacher, Andrea Pietracaprina, Geppino Pucci, Eli Upfal, and Fabio Vandin. An efficient rigorous approach for identifying statistically significant frequent itemsets. *Journal of the ACM (JACM)*, 59(3):1–22, 2012.

Junpei Komiyama, Masakazu Ishihata, Hiroki Arimura, Takashi Nishibayashi, and Shin-ichi Minato. Statistical emerging pattern mining with multiple testing correction. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 897–906. ACM, 2017.

Kleanthis-Nikolaos Kontonasios, Eirini Spyropoulou, and Tijl De Bie. Knowledge discovery interestingness measures based on unexpectedness. *WIREs Data Mining and Knowledge Discovery*, 2(5):386–399, 2012.

doi: https://doi.org/10.1002/widm.1063. URL https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1063.

Felipe Llinares-López, Mahito Sugiyama, Laetitia Papaxanthos, and Karsten Borgwardt. Fast and memory-efficient significant pattern mining via permutation testing. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 725–734. ACM, 2015.

Cécile Low-Kam, Chedy Raïssi, Mehdi Kaytoue, and Jian Pei. Mining statistically significant sequential patterns. In *2013 IEEE 13th International Conference on Data Mining*, pages 488–497. IEEE, 2013.

Nimrod Megiddo and Ramakrishnan Srikant. Discovering predictive association rules. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, KDD '98, pages 274–278, 1998.

Shin-ichi Minato, Takeaki Uno, Koji Tsuda, Aika Terada, and Jun Sese. A fast method of statistical assessment for combinatorial hypotheses based on frequent itemset enumeration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 422–436. Springer, 2014.

Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

Markus Ojala. Assessing data mining results on matrices with randomization. In *2010 IEEE International Conference on Data Mining*, pages 959–964, 2010. doi: 10.1109/ICDM.2010.20.

Markus Ojala, Niko Vuokko, Aleksi Kallio, Niina Haiminen, and Heikki Mannila. Randomization of real-valued matrices for assessing the significance of data mining results. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, SDM '08, pages 494–505, 2008. doi: 10.1137/1.9781611972788.45. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972788.45.

Markus Ojala, Gemma C. Garriga, Aristides Gionis, and Heikki Mannila. Evaluating query result significance in databases via randomizations. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM)*, pages 906–917, 2010. doi: 10.1137/1.9781611972801.79.

Leonardo Pellegrina and Fabio Vandin. Efficient mining of the most significant patterns with permutation testing. *Data Mining and Knowledge Discovery*, 34:1201–1234, 2020.

Leonardo Pellegrina, Matteo Riondato, and Fabio Vandin. SPuManTE: Significant pattern mining with unconditional testing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1528–1538, New York, NY, USA, 2019a. ACM. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3330978. URL http://doi.acm.org/10.1145/3292500.3330978.

Leonardo Pellegrina, Matteo Riondato, and Fabio Vandin. Hypothesis testing and statistically-sound pattern mining. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 3215–3216, New York, NY, USA, 2019b. ACM. ISBN 978-1-4503-6201-6. doi: 10.1145/3292500.3332286. URL http://doi.acm.org/10.1145/3292500.3332286.

Richard P. Stanley. *Enumerative Combinatorics*, volume 1. Cambridge University Press, 2 edition, 2011.

Pan-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right objective measure for association analysis. *Inf. Sys.*, 29:293–313, 2004.

Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences*, 110(32):12996–13001, 2013.

Andrea Tonon and Fabio Vandin. Permutation strategies for mining significant sequential patterns. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1330–1335. IEEE, 2019. Full version at https://www.dei.unipd.it/~vandinfa/ICDM19full.pdf.

Jilles Vreeken and Nikolaj Tatti. Interesting patterns. In *Frequent pattern mining*, pages 105–134. Springer, 2014.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2005.

Geoffrey I. Webb. Discovering significant rules. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 434–443. ACM, 2006.

Geoffrey I. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, 2007.

Geoffrey I Webb. Layered critical values: a powerful direct-adjustment approach to discovering significant patterns. *Machine Learning*, 71(2-3):307–323, 2008.

Peter H. Westfall and Stanley S. Young. *Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment*. Wiley-Interscience, 1993.

Jun Wu, Zengyou He, Feiyang Gu, Xiaoqing Liu, Jianyu Zhou, and Can Yang. Computing exact permutation p-values for association rules. *Information Sciences*, 346:146–162, 2016.