Data Mining and Knowledge Discovery manuscript No.
(will be inserted by the editor)

SPEck: Mining Statistically-significant Sequential Patterns Efficiently with Exact Sampling

Steedman Jenkins ·
Stefan Walzer-Goldfeld ·
Matteo Riondato

Received: 8 October 2021 / Accepted: 7 June 2022

Abstract We study the problem of efficiently mining statistically-significant sequential patterns from large datasets, under different null models. We consider one null model presented in the literature, and introduce two new ones that preserve different properties of the observed dataset. We describe SPEck, a generic framework for significant sequential pattern mining, that can be instantiated with any null model, when given a procedure for sampling datasets according to the null distribution. For the previously-proposed model, we introduce a novel procedure that samples *exactly* according to the null distribution, while existing procedures are *approximate* samplers. Our exact sampler is also more computationally efficient and much faster in practice. For the null models we introduce, we give exact and/or almost uniform samplers. Our experimental evaluation shows how exact samplers can be orders of magnitude faster than approximate ones, and scale well.

Keywords Hypothesis Testing, Significant Pattern Mining, Statistically-Sound Knowledge Discovery

Speck Alto Adige / Südtiroler Speck: an Italian dry-cured, lightly smoked ham.

1 Introduction

Representing data as sets of sequences of elements is natural for many processes. Finding *interesting* patterns in a dataset of sequences is an important knowledge discovery task with many applications: web log analysis, finance modelling, monitoring of athletes' vitals and performance (Hrovat et al. 2015), and processing of satellite images (Méger et al. 2015). As already observed for other kinds of patterns (Hämäläinen and Webb 2019; Pellegrina et al. 2019a), support (or frequency), i.e., how many times a pattern appears in a dataset, falls short from being a good measure of interestingness: a sequential pattern may be frequent just because it is composed of many frequent items, but it may not be interesting in itself. For this

Steedman Jenkins and Stefan Walzer-Goldfeld contributed equally to this work.

S. Jenkins, S. Walzer-Goldfeld, M. Riondato
Dept. of Computer Science, Amherst College, Amherst, MA, USA
E-mail: {sjenkins23,swalzergoldfeld23,mriondato}@amherst.edu

reason, many other measures of interestingness have been proposed, e.g., based on Minimum Description Length (Lam et al. 2014) or on different statistical assumptions (Fumarola et al. 2016; Feremans et al. 2018; Petitjean et al. 2016; Tatti 2015; Raïssi et al. 2008; Gwadera and Crestani 2010; Low-Kam et al. 2013; Tonon and Vandin 2019; Pinxteren and Calders 2021). A natural way to find interesting sequential patterns is to perform statistical hypothesis tests on the patterns, under the assumption of a user-specified *null model* that captures properties of the data generating process as expressed in the observed dataset (see Sect. 3.1 for formal definitions). Sequences that "pass the test" are deemed interesting, and marked as *(statistically-)significant*. It is of key importance that the testing procedure corrects for the fact that *multiple hypotheses* are being tested, which is usually done by ensuring that the *Family-Wise Error Rate (FWER)*—i.e., the probability that the collection of patterns marked as significant contains a pattern that is not really significant—is bounded by a user-specified quantity $\delta \in (0, 1)$.

Different from, e.g., the collection of frequent sequences, the set of significant sequential patterns is *not uniquely defined*: it depends on the *null model* that the user *chooses* to assume. Many null models for sequential patterns have been defined in the literature (see Sect. 2). The choice of a null model must be *very deliberate*, and depends on what is known (and should be modeled) about the process that generates data. Informally, finding the collection of significant sequential patterns can lead to the discovery of properties of the data generation process that are not captured by the assumed model. It is therefore important to have *efficient* methods for significant sequential pattern mining that can *handle different null models*.

A key step in hypothesis testing for sequences is the computation of the *p-value* (Sect. 3.1), i.e., the probability that, under the null model, the support of the pattern is as high or higher than in the observed dataset. Computing the p-value exactly is usually impossible, except in restricted cases (Pinxteren and Calders 2021). Thus, algorithms for significant sequence mining rely on *estimating* p-values by *randomly sampling datasets* from the null model. Sampled datasets are also used in Monte-Carlo methods such as the Westfall-Young approach (Westfall and Young 1993) for computing the adjusted critical value used for controlling the FWER (see Sect. 3.1). It is therefore crucially important to have *efficient procedures for sampling datasets from the null model*, where efficiency is considered both along the axis of computation (in terms of the time complexity to generate a dataset) and along the axis of probability, i.e., *how close to the desired null distribution* is the output distribution of the sampling procedures: *exact samplers* that are computationally efficient should be preferred to samplers whose outputs *approximately* follow the null distribution.

Contributions We study the problem of mining statistically significant sequential patterns from a large transactional dataset, with statistical guarantees and as efficiently as possible. Our contributions are the following.

- We present SPEck (Alg. 1), a generic framework for mining significant sequential patterns according to different null models. Our framework uses the Westfall-Young resampling approach (Westfall and Young 1993), and generalizes the ProMiSe algorithm by Tonon and Vandin (2019).
- SPEck's instantiation for a null model requires a procedure to sample datasets according to the null distribution. For a popular null model (Sect. 4.2.1), previous work (Tonon and Vandin 2019) gave an *approximate* sampling procedure

(specifically, an ε-Almost Uniform Sampler (ε-AUS), see Sect. 3.2). We introduce the first *Exact Uniform Sampler (EUS)* for this null model. In addition to being better on *probabilistic* grounds, our EUS is computationally more efficient than the ε-AUS, thus it should be preferred. As a byproduct of our approach, we also show an improved mixing time for the ε-AUS by Tonon and Vandin (2019).
- We focus on market basket data, i.e., binary sequential datasets (see Sect. 3), but most of what we say can be extended to richer sequential datasets, such as those used for mining high-utility sequential patterns (Truong-Chi and Fournier-Viger 2019).
- We introduce two novel null models (Sect. 4.2.2 and 4.2.3) preserving different properties of the observed dataset, and we give EUS's and/or ε-AUS's for each of them. This contribution enriches the set of null models available to practitioners.
- We implement SPEck and our EUS's and ε-AUS's, and evaluate their performance on real and artificial datasets (Sect. 5) in terms of the time needed to sample a dataset from the null model and the time to mine significant patterns. The results show that our EUS's are faster than ε-AUS's for the same null model, by up to 26 times, and thus that they should be preferred for the task at hand.

2 Related Work

Frequency/support was the first interestingness measure for sequences (Agrawal and Srikant 1995), and efficient algorithms are readily available, both exact (Pei et al. 2004; Fournier-Viger et al. 2014) and approximate (Raïssi and Poncelet 2007; Servan-Schreiber et al. 2020). The limitations of this measure were soon understood, and the knowledge discovery community shifted its focus to developing other ways of assessing the interestingness of sequential patterns, based, for example, on Minimum Description Length (Lam et al. 2014), exceptional model mining (Mollenhauer and Atzmueller 2020) or on different statistical models (Fumarola et al. 2016; Feremans et al. 2018; Petitjean et al. 2016; Tatti 2015; Raïssi et al. 2008; Gwadera and Crestani 2010; Low-Kam et al. 2013; Tonon and Vandin 2019; Pinxteren and Calders 2021). This shift is similar to that observed for other kinds of patterns (e.g., itemsets (Gionis et al. 2007; Pellegrina et al. 2019b), subgraphs (Sugiyama et al. 2015)), and subgroups (Duivesteijn and Knobbe 2011). We refer the reader to the survey by Hämäläinen and Webb (2019) and the tutorial by Pellegrina et al. (2019a) for in-depth treatments of this field. In the interest of brevity and clarity, we discuss here only works dealing with sequential patterns in an *unlabeled* dataset, i.e., when the transactions are not associated to a class label (Pellegrina and Vandin (2020) discuss the labeled setting).

Gwadera and Crestani (2010) propose a two-parts probabilistic model for the sequences in the dataset, based on a model for just the lengths of the sequences and a maximum-entropy null model for the itemsets in the sequences. Testing for significance this way is particularly inefficient for longer patterns. Our approach does not suffer from this issue. Additionally, Gwadera and Crestani's model does not exactly preserve some important properties of the observed dataset, which is a desirable feature of null models. All null models we study (Sect. 4.2) exactly preserve one or more properties, e.g., the multi-set of transaction lengths or their item-lengths, or the multi-support of the itemsets (see definitions in Sect. 3).

Low-Kam et al. (2013) introduce SigSpan, an algorithm for mining statistically significant sequential patterns. In their model, each itemset appears in a transaction with a probability equal to its frequency in the original dataset, and independently from any other event. The frequency is therefore preserved only in expectation. The null models we study preserve exactly other important properties of the original dataset. Additionally, Low-Kam et al. use the Bonferroni correction (Bonferroni 1936) to control for the Family-Wise Error Rate (FWER, see Sect. 3) and only consider the frequent patterns in the observed dataset as the set of hypotheses. We instead use the resampling approach by Westfall and Young (1993), which tends to have more statistical power because it takes into consideration the correlation between the tested hypotheses (i.e., patterns). Furthermore, our approach considers the class of all patterns that may be frequent in *any* dataset from the null model, which is statistically more appropriate.

Tonon and Vandin (2019) present ProMiSe, an algorithm that mines significant frequent sequential patterns under two different null models while controlling the FWER using the Westfall-Young Monte-Carlo resampling approach. SPEck generalizes this algorithm. For one of the null models they propose, we show an *exact* sampler, which is preferable both probabilistically and computationally to the *approximate* sampler they propose. We also introduce two novel null models that are not considered in their work.

Pinxteren and Calders (2021) have developed PS^2, an algorithm to rank the significance of sequential patterns according to the specific null model that considers only permutations of itemsets *within* a transaction, not *across* different transactions. This model was previously presented by Tonon and Vandin (2019). PS^2 only works in the case where all itemsets in a transaction have length one and there are no repeated itemsets in a transaction. These assumptions are made to avoid what they call *multiple-distribution sensitivity*, the assumption that all sequences come from the same distribution. By considering all permutations of each sequence, PS^2 computes the fraction of permutations in which a sequential pattern appears, thus allowing for the exact computation of the p-value. We study different null models, without imposing the above restrictions, but the spirit of our work is similar in the sense that we are developing efficient methods for computing (approximate) p-values of patterns under interesting null models.

3 Preliminaries

We now formally define the main concepts used in this work. To keep the presentation focused, we only cover the case of binary sequential datasets, but what we say can also be extended to the richer datasets used for high-utility sequence mining (Truong-Chi and Fournier-Viger 2019). A *ground set* (or *alphabet*) $\mathcal{A}$ is a finite set of *items* $\mathcal{A} \doteq \{a_1, \ldots, a_n\}$, for some $n > 1$. An *itemset* $A \subseteq \mathcal{A}$ is a non-empty subset of $\mathcal{A}$. A *sequential pattern*, or *sequence* $S = \langle A_1, \ldots, A_\ell \rangle$ for some $\ell \geq 1$, is a finite *ordered* list of itemsets, i.e., $A_i \subseteq \mathcal{A}$, $1 \leq i \leq \ell$. We say that these itemsets *participate* in the sequence and denote this fact with $A_i \in S$. The itemset A may be repeated more than once in a sequence (e.g., $S = \langle A, B, A \rangle$). The *length* $|S|$ of a sequence S is the number of itemsets participating in S, and the *item-length* $\|S\|$ of S is defined as $\sum_{A \in S} |A|$. For example, $S = \langle \{a, b\}, \{b, c, d\}, \{a, f\} \rangle$ has length $|S| = 3$ and item-length $\|S\| = 7$. A sequence $S = \langle S_1, S_2, \ldots, S_y \rangle$ is a *subsequence* of a

sequence $T = \langle T_1, T_2, \ldots, T_w \rangle$ (denoted $S \sqsubseteq T$) iff there exist integers $1 \leq i_1 < i_2 < \cdots < i_y \leq w$ such that $S_1 \subseteq T_{i_1}, S_2 \subseteq T_{i_2}, \ldots, S_y \subseteq T_{i_y}$. A *transactional dataset* $\mathcal{D}$ is a bag of sequences. Each sequence in $\mathcal{D}$ is also called, in this context, a *transaction*. The *support* $\sigma_{\mathcal{D}}(S)$ *of a sequence* S *in* $\mathcal{D}$ is the *number* of transactions in $\mathcal{D}$ of which S is a subsequence. The *support* $\sigma_{\mathcal{D}}(A)$ *of an itemset* A *in* $\mathcal{D}$ is the number of transactions t of D in which A participates, i.e., for which $A \in t$. The *multi-support* $\rho_{\mathcal{D}}(A)$ *of an itemset* A *in* $\mathcal{D}$ is the number of *times* that A is repeated in total in the transactions of $\mathcal{D}$. For example, if $\mathcal{D} = \{\langle A, B \rangle, \langle A, C, A \rangle, \langle B, C \rangle\}$, it holds $\sigma_{\mathcal{D}}(A) = 2$ and $\rho_{\mathcal{D}}(A) = 3$.

Let $\mathbb{S}$ be the (infinite) set of all possible sequences built on itemsets with ground set $\mathcal{A}$. Given a *minimum support threshold* $\theta \in [1, |\mathcal{D}|]$ (where $|\mathcal{D}|$ is the number of transactions in $\mathcal{D}$), the set $\mathsf{F}_{\mathcal{D}}(\theta)$ of θ*-frequent sequences in* $\mathcal{D}$ is the set of sequences with support at least θ in $\mathcal{D}$, i.e.,

$$\mathsf{F}_{\mathcal{D}}(\theta) \doteq \{S \in \mathbb{S} \ : \ \sigma_{\mathcal{D}}(S) \geq \theta\} \ .$$

3.1 Significant Patterns and Hypothesis Testing

Given an *observed dataset* $\mathcal{D}$, whose transactions are built on the alphabet $\mathcal{A}$, a *null model* Π is a pair $\Pi \doteq (\mathcal{Z}, \pi)$ where $\mathcal{Z}$ is a *subset* of the set of all possible datasets with $|\mathcal{D}|$ transactions built on $\mathcal{A}$, and π is a probability distribution over $\mathcal{Z}$. In this work, it is always the case that π is the *uniform* distribution over $\mathcal{Z}$, i.e., $\pi \doteq \mathcal{U}(\mathcal{Z})$. The set $\mathcal{Z}$ is usually defined by including all and only the datasets as above that exhibit some property that $\mathcal{D}$ also exhibits (thus, $\mathcal{D} \in \mathcal{Z}$). For example, $\mathcal{Z}$ may contain all and only the datasets for which $\rho_{\mathcal{D}'}(A) = \rho_{\mathcal{D}}(A)$, for every $\mathcal{D}' \in \mathcal{Z}$, and every itemset $A \subseteq \mathcal{A}$ (i.e., the multi-support of every itemset is the same in all datasets in $\mathcal{Z}$). In Sect. 4, we study several null models preserving different properties.

Given a null model $\Pi = (\mathcal{Z}, \pi)$, the *expected support* $\mu_{\Pi}(S)$ *of a sequence* $S \in \mathbb{S}$ *under* Π is the expectation of the support of S w.r.t. π, i.e.,

$$\mu_{\Pi}(S) \doteq \mathbb{E}_{\mathcal{D} \sim \pi}\left[\sigma_{\mathcal{D}}(S)\right] \ .$$

We restrict $\pi = \mathcal{U}(\mathcal{Z})$, and $\mathcal{Z}$ is finite, thus we can equivalently write

$$\mu_{\Pi}(S) = \frac{1}{|\mathcal{Z}|} \sum_{\mathcal{D} \in \mathcal{Z}} \sigma_{\mathcal{D}}(S) \ .$$

In this work, given a dataset $\mathcal{D}$, a minimum support threshold θ, and a null model $\Pi = (\mathcal{Z}, \pi)$, we are interested in finding a subset of $\mathsf{F}_{\mathcal{D}}(\theta)$ containing only sequences whose support in $\mathcal{D}$ is *significantly* different than their expected support under Π, where significance is determined using *hypothesis testing*. Specifically, for each sequence S, we consider the null hypothesis

$$H_S \doteq \text{“}\mu_{\Pi}(S) = \sigma_{\mathcal{D}}(S)\text{”} \ .^{1}$$

The *p-value* $\mathrm{p}_{\mathcal{D}'}(S)$ *of* S *in* $\mathcal{D}'$ is the probability, conditioned on H_S (i.e., under Π), that, in a dataset $\mathcal{D}''$ drawn from $\mathcal{Z}$ according to π, one observes $\sigma_{\mathcal{D}''}(S) \geq \sigma_{\mathcal{D}'}(S)$,

[1] This null hypothesis is one of many that could be considered: one could equally test any hypothesis for which the pattern support (or actually any pattern-specific function of the dataset) is a reasonable test statistic.

i.e., a support of S at least as large as its support in $\mathcal{D}'$. The p-value $\mathrm{p}_{\mathcal{D}}(S)$ is used to assess whether the observed dataset $\mathcal{D}$ gives evidence for the null hypothesis H_S to be false: informally, a small $\mathrm{p}_{\mathcal{D}}(S)$ (e.g., not larger than a *critical value* α) is taken as suggesting that there is such evidence. When the p-value $\mathrm{p}_{\mathcal{D}}(S)$ is such that this evidence seems present, the null hypothesis H_S is *rejected* and the sequence S is marked as *significant*. The p-value of S in $\mathcal{D}$ may be low even if the null hypothesis is true, thus there is the possibility that marking S as significant is a *false discovery*, or, equivalently, that S is a *false positive*. For example, if we decide to mark S as significant whenever $\mathrm{p}_{\mathcal{D}}(S)$ is not larger than a critical value α, then the probability (considered over repetitions of the experiment, i.e., over different observed datasets) to make a false discovery involving S is at most α.

Under the null models we consider, computing the p-value $\mathrm{p}_{\mathcal{D}'}(S)$ *exactly* is not possible: even if π is *uniform* over $\mathcal{Z}$, the support of S, seen as a random variable, has a complicated distribution. We then *estimate* the p-value $\mathrm{p}_{\mathcal{D}'}(S)$ using the following Monte-Carlo procedure (Tonon and Vandin 2019, Sect. II.D). Let $\mathcal{D}_1, \ldots, \mathcal{D}_T$ be T datasets sampled independently from π. We estimate $\mathrm{p}_{\mathcal{D}'}(S)$ as the fraction $\tilde{\mathrm{p}}_{\mathcal{D}'}(S)$ of the datasets $\mathcal{D}', \mathcal{D}_1, \ldots, \mathcal{D}_T$ where the support of S was at least $\sigma_{\mathcal{D}'}(S)$, i.e.,

$$\tilde{\mathrm{p}}_{\mathcal{D}'}(S) \doteq \frac{1}{T+1}\left(1 + \sum_{i=1}^{T} \mathbb{1}[\sigma_{\mathcal{D}_i}(S) \geq \sigma_{\mathcal{D}'}(S)]\right), \tag{1}$$

where $\mathbb{1}[\cdot]$ is the indicator function for the condition between brackets.

In this work we develop methods for finding a subset Q of $\mathsf{F}_{\mathcal{D}}(\theta)$ such that the probability that *any* sequence in Q is a false positive is at most δ, for a user-specified parameter $\delta \in (0, 1)$. In statistical terms, we want to develop methods that output a set Q while controlling the *Family-Wise Error Rate* (FWER) at level δ.

A classic way to control the FWER at level δ is the *Bonferroni correction* (Bonferroni 1936) (later slightly improved by many, e.g., Holm (1979)). The idea is that, when testing a set $\mathcal{H}$ of hypotheses, one should use the *adjusted critical value* $\alpha_{\mathrm{B}}(\delta, \mathcal{H}) \doteq \delta/|\mathcal{H}|$ to test each hypothesis (i.e., compare each p-value to α_{B}). This approach suffers from many defects: *(1)* as the number k of hypotheses grows, it becomes harder to reject false null hypotheses, i.e., this approach suffers from *low statistical power*; *(2)* it does not take into account any "structure" or correlation between the different hypotheses; and *(3)*, it cannot be applied when the *number* of hypotheses being tested is not known *in advance*, or is *infinite*. The first two issues affect all applications of the Bonferroni approach, and already make it quite unattractive, but it is the third one that really prevents us from using this technique to control the FWER for the task in which we are interested. Indeed, under the null model, we have to consider the set $\mathcal{H}$ of the hypotheses associated with *all* sequences S for which there exists a $\mathcal{D}' \in \mathcal{Z}$ such that $\sigma_{\mathcal{D}'}(S) \geq \theta$ (Tonon and Vandin 2019, Sect. I.B). While the *set* of these hypotheses is well defined, its size is not readily available, thus precluding the use of the Bonferroni correction for this task.

The *Westfall-Young* method (Westfall and Young 1993) for multiple hypothesis correction relies on an *empirical* approximation of the distribution of the *minimum* p-value across all sampled datasets to determine an adjusted critical value $\alpha_{\mathrm{WY}}(\delta, \mathcal{H})$ as follows. Let $\mathcal{D}_1, \ldots, \mathcal{D}_P$ be P datasets sampled independently from π. Let

$$\check{p}_i \doteq \min\{\mathrm{p}_{\mathcal{D}_i}(S) \ : \ H_S \in \mathcal{H}\}, 1 \leq i \leq P \tag{2}$$

be the *minimum* p-value on $\mathcal{D}_i$ of any null hypothesis in $\mathcal{H}$. Then, the Westfall-Young adjusted critical value is

$$\alpha_{\mathrm{WY}}(\delta, \mathcal{H}) \doteq \max \left\{ \alpha \ : \ \frac{1}{P} \sum_{i=1}^{P} \mathbb{1}[\check{p}_i \leq \alpha] \leq \delta \right\} \ . \tag{3}$$

Except in restricted settings (Pinxteren and Calders 2021), $\mathrm{p}_{\mathcal{D}_i}(S)$ cannot be easily computed, so $\tilde{\mathrm{p}}_{\mathcal{D}_i}(S)$ can be used in its place in (2) (Tonon and Vandin 2019).

3.2 Uniform and ε-Almost-Uniform Samplers

In this work we discuss procedures to draw samples from a finite domain Ω, according to a distribution π. To be more precise, the procedures take an input x from some space, and draw a sample from a finite domain $\Omega(x)$ that depends on x, based on π. In the specific case of significant sequential pattern mining, x is the observed dataset $\mathcal{D}$, and $\Omega(x)$ is the set $\mathcal{Z}$ of datasets considered in the null model $\Pi = (\mathcal{Z}, \pi)$, where, as mentioned in Sect. 3.1, π is uniform over $\mathcal{Z}$, and $\mathcal{Z}$ depends on the observed dataset $\mathcal{D}$ because the datasets in $\mathcal{Z}$ preserve specific properties of $\mathcal{D}$.

We study two kinds of sampling procedures: *exact-π samplers* and *ε-almost-π samplers*. Since in this work π will always be the uniform distribution over $\Omega(x)$, we actually talk about Exact Uniform Samplers (EUS) and ε-Almost Uniform Samplers (ε-AUS). EUS's, as the name implies, are algorithms that, given $\Omega(x)$ return a sample from $\Omega(x)$ that is distributed perfectly uniformly in $\Omega(x)$. We only consider EUS's that run in time polynomial in the size of the input x.

An ε-AUS (Mitzenmacher and Upfal 2005, Sect. 10.3) is instead an algorithm A that takes in input x and a parameter $\varepsilon \in (0, 1)$, and outputs a sample w from $\Omega(x)$ such that the *total variation distance* $\mathsf{d}(\psi_{\mathsf{A}}, \pi)$ between π and the distribution ψ_{A} of the output w, when A is run with input x and parameter ε, is at most ε, i.e.,

$$\mathsf{d}(\psi_{\mathsf{A}}, \pi) \doteq \max_{S \subseteq \Omega(x)} \left| \psi_{\mathsf{A}}(S) - \underbrace{\pi(S)}_{\doteq \frac{|S|}{|\Omega(x)|}} \right| \leq \varepsilon \ .$$

We only consider ε-AUS's that run in time polynomial in $\ln \varepsilon^{-1}$ and in the size of x (also known as "Fully-Polynomial ε-AUS's").

It should be evident that an EUS is always preferable to an ε-AUS in probabilistic terms, but the computational complexity must also be taken into consideration. In this work we show that it is possible to develop EUS's that are faster than ε-AUS's, thus making the former the obvious choice for the task at hand.

Markov-Chain-Monte-Carlo ε-AUS and mixing times Many ε-AUS's are based on Markov-Chain-Monte-Carlo (MCMC) methods (Mitzenmacher and Upfal 2005, Ch. 10). These methods use a Markov chain whose states are the elements of the domain $\Omega(x)$ and whose unique stationary distribution is the uniform distribution. Samples are taken by running the Markov chain *long enough* that the distribution of the state of the Markov chain has a total-variation distance at most ε from the uniform distribution. The number of steps needed for this condition to hold may depend on

the starting state $s \in \Omega(x)$ from which the chain starts. For any $s \in \Omega(x)$, let p_s^t be the distribution of the state of the chain starting at s after t steps. The *mixing time* $\tau(\varepsilon)$ for a Markov chain is the minimum number of steps t such that the maximum, over the choice of the starting state s, of the total variation distance between p_s^t and the stationary distribution π of the chain is at most ε:

$$\tau(\varepsilon) \doteq \min \left\{ t \ : \ \max_{s \in \Omega(x)} \mathsf{d}(p_s^t, \pi) \leq \varepsilon \right\} \ .$$

A Markov chain is said to be rapidly mixing if the mixing time is polynomial in $\log(1/\varepsilon)$ and in the size of x. It is important to remark that $\tau(\varepsilon)$ is a function of ε, and, unless $\varepsilon = 0$, it is *not* the time needed for the distribution of the state of the chain to be *exactly* the stationary distribution. That is, $\tau(\varepsilon)$ is not a *strong stationary time* (Levin and Peres 2017, Ch. 6) (which is a random variable), nor in general an upper bound to such a time. Many techniques exist to bound the mixing time $\tau(\varepsilon)$, for example coupling and path coupling which give upper bounds that depend on *both* ε and the size of x. Often, in the derivation of such bounds, ε is fixed to a constant, e.g., $1/4$, as $\tau(\varepsilon) \leq \lceil \log_2 \varepsilon^{-1} \rceil \tau(1/4)$ (Levin and Peres 2017, Eq. 4.34). The monograph by Levin and Peres (2017) contains an in-depth discussion of Markov chains, mixing and (strong) stationary times, and (path) coupling.

4 Mining Significant Sequential Patterns Efficiently

In this section we first describe SPEck, our framework for mining significant sequential patterns, and then (Sect. 4.2) discuss different null models, some of them novel, and show procedures to efficiently sample from them.

4.1 SPEck

We now present SPEck (pseudocode in Alg. 1), a generic framework for mining significant sequential patterns, using the Westfall-Young approach. The framework closely follows ProMiSe by Tonon and Vandin (2019, Alg. 2), but we make it capable of handling any null model by making the null model part of the input, (see below) and we make SPEck use a single set of T datasets to estimate all the p-values, rather than having to sample a new set every time. This last change allows for a higher level of parallelism when the algorithm is implemented, and makes no difference from a correctness point of view.

The input parameters are: a dataset $\mathcal{D}$, a minimum support threshold $\theta \in [1, |\mathcal{D}|]$, a null model $\Pi = (\mathcal{Z}, \pi)$, an acceptable FWER $\delta \in (0, 1)$, and two integers T and P, that respectively specify the number of datasets from Π to use for estimating p-values, and the number of datasets from Π to use for computing the adjusted critical value α_{WY}. The output is a collection $Q \subseteq \mathsf{F}_{\mathcal{D}}(\theta)$, with the following properties.

Lemma 1 *With probability at least* $1-\delta$ *(over the runs of the algorithm),* Q *contains* only *sequences* S *for which the null hypothesis* H_S *(w.r.t.* Π*) is false.*

Equivalently, SPEck controls the FWER at level δ. The proof of this result is immediate from the description and the correctness of the Westfall-Young approach.

Algorithm 1: SPEck: a framework for mining significant sequential patterns

Input : Dataset $\mathcal{D}$, min. supp. thres. $\theta \in [1, |\mathcal{D}|]$, null model $\Pi = (\mathcal{Z}, \pi)$, acceptable FWER $\delta \in (0, 1)$, no. of datasets for p-value estimation T, no. of datasets for critical value computation P

Output: Set $Q \subseteq \mathsf{F}_{\mathcal{D}}(\theta)$, with no false positive, with prob. $\geq 1 - \delta$

1 $\mathcal{F} \leftarrow$ array of size T;
2 **for** $i \leftarrow 1$ **to** T **do**
3 $\quad \mathcal{D}_e^{(i)} \leftarrow$ `sampleDataset`($\mathcal{D}$, Π);
4 $\quad \mathcal{F}[i] \leftarrow \mathsf{F}_{\mathcal{D}_e^{(i)}}(\theta)$;
5 $\mathcal{P} \leftarrow$ array of size P of all 1's;
6 **for** $i \leftarrow 1$ **to** P **do**
7 $\quad \mathcal{D}_c^{(i)} \leftarrow$ `sampleDataset`($\mathcal{D}$, Π);
8 $\quad$ **foreach** $S \in \mathsf{F}_{\mathcal{D}_c^{(i)}}(\theta)$ **do**
9 $\quad\quad \tilde{\mathsf{p}}_{\mathcal{D}_c^{(i)}}(S) \leftarrow$ `estimatePValue`($\mathcal{F}$, $\sigma_{\mathcal{D}_c^{(i)}}(S)$);
10 $\quad\quad \mathcal{P}[i] \leftarrow \min\left\{\mathcal{P}[i], \tilde{\mathsf{p}}_S(\mathcal{D}_c^{(i)})\right\}$;
11 $\alpha_{\mathrm{WY}} \leftarrow$ `adjustCriticalValue`($\mathcal{P}$, δ);
12 $Q \leftarrow \emptyset$;
13 **foreach** $S \in \mathsf{F}_{\mathcal{D}}(\theta)$ **do**
14 $\quad \tilde{\mathsf{p}}_{\mathcal{D}}(S) \leftarrow$ `estimatePValue`($\mathcal{F}$, $\sigma_{\mathcal{D}}(S)$);
15 $\quad$ **if** $\tilde{\mathsf{p}}_{\mathcal{D}}(S) \leq \alpha_{\mathrm{WY}}$ **then** $Q \leftarrow Q \cup \{S\}$;
16 **return** Q

We assume SPEck has access to a standard frequent sequence mining algorithm (Pei et al. 2004; Fournier-Viger et al. 2014). It starts by creating an array $\mathcal{F}$ of size T (line 1), initially empty. This array is populated so that its i-th element, $1 \leq i \leq T$, is the collection of θ-frequent sequences of a dataset $\mathcal{D}_e^{(i)}$ sampled independently from Π (lines 3–4). The collection $\mathcal{F}$ is used later in the algorithm for estimating the p-values of patterns, as in (1). The function `sampleDataset` samples a single dataset from $\mathcal{Z}$. It takes $\mathcal{D}$ as input because $\mathcal{Z}$ depends on $\mathcal{D}$ (Sect. 3.2). SPEck then creates an array $\mathcal{P}$ of size P, initialized to contain all ones (line 5). This array will store samples from the minimum p-value distribution, from which the adjusted critical value α_{WY} is computed. This value is then used to determine the patterns to include in Q. The value in $\mathcal{P}[i]$, $1 \leq i \leq P$ is computed as follows. First, a dataset $\mathcal{D}_c^{(i)}$ is sampled from Π (line 7). Then, the array $\mathcal{F}$ is used to estimate the p-value on $\mathcal{D}_c^{(i)}$ of each frequent sequence $S \in \mathsf{F}_{\mathcal{D}_c^{(i)}}(\theta)$ (lines 8–9). If the estimated p-value $\tilde{\mathsf{p}}_{\mathcal{D}_c^{(i)}}(S)$ is less than $\mathcal{P}[i]$, this latter quantity is updated to be $\tilde{\mathsf{p}}_{\mathcal{D}_c^{(i)}}(S)$ (line 10). This way, $\mathcal{P}[i]$ will contain the minimum estimated p-value on $\mathcal{D}_c^{(i)}$ among all θ-frequent sequences in $\mathcal{D}_c^{(i)}$. Once all the elements of $\mathcal{P}$ have been computed, the algorithm obtains the adjusted critical value α_{WY} as in (2) (line 11). SPEck then mines the original dataset $\mathcal{D}$ and, for each $S \in \mathsf{F}_{\mathcal{D}}(\theta)$, it estimates its p-value using $\mathcal{F}$ (line 14). If this quantity is not larger than the adjusted critical value, S is added to the output set Q (line 15), which is returned at the end.

SPEck offers many possibilities for parallelization, in a way similar to the ProMiSe algorithm (Tonon and Vandin 2019, Sect. III.B).

4.2 Sampling datasets from the null model

SPEcK requires a procedure (function `sampleDataset` in Alg. 1) to sample a dataset from the null model Π. We now discuss how to perform this operation efficiently in both computational and "probabilistic" terms, for different null models. For one of the null models we study (Sect. 4.2.1), an ε-AUS (Sect. 3.2) was proposed in the literature (Tonon and Vandin 2019). We introduce an *EUS* for this model, which is also much faster than the existing ε-AUS. We also discuss other null models that we deem particularly interesting and have not been studied before, and give EUS's and/or ε-AUS's for them (Sect. 4.2.2 and 4.2.3).

4.2.1 Null model #1: fixed multi-supports and transaction lengths

We start by studying a null model $\Pi = (\mathcal{Z}, \pi)$ proposed by Tonon and Vandin (2019, Sect. II.B), and for which they give an ε-AUS (see below). This null model can be seen as an extension of the null model proposed by Gionis et al. (2007) from itemsets to sequential patterns. Given the observed dataset $\mathcal{D}$, fix an *arbitrary ordering* $t_1, \ldots, t_{|\mathcal{D}|}$ of its transactions. The null set $\mathcal{Z}$ is the set of all datasets $\mathcal{D}'$ with $|\mathcal{D}'| = |\mathcal{D}|$ transactions such that:

1. the transactions $t'_1, \ldots, t'_{|\mathcal{D}|}$ of $\mathcal{D}'$ are such that $|t'_i| = |t_i|$, $1 \leq i \leq |\mathcal{D}|$, i.e., the lengths of the transactions of $\mathcal{D}$, as imposed by the fixed arbitrary ordering, are preserved; and
2. for every itemset A participating in at least one transaction t, it holds $\rho_{\mathcal{D}'}(A) = \rho_{\mathcal{D}}(A)$, i.e., the multi-supports of itemsets participating in the transactions are preserved.[2]

The distribution π is uniform over $\mathcal{Z}$, as it is always the case in this work. It is important to remark that in this model, two *different* datasets satisfying both the above requirements may differ from each other only in the *ordering* (in the datasets) of transactions of the same length, even if both datasets are identical *bags* of transactions. For example, the following two datasets

$$\mathcal{D} \doteq \begin{matrix} \langle C, A \rangle \\ \langle B \rangle \\ \langle A \rangle \end{matrix} \qquad \text{and} \qquad \mathcal{D}' \doteq \begin{matrix} \langle C, A \rangle \\ \langle A \rangle \\ \langle B \rangle \end{matrix}, \tag{4}$$

where A, B, and C are itemsets, are *different* datasets, both in $\mathcal{Z}$. Without this assumption, i.e., if we consider the two above datasets the same dataset, the MCMC algorithm proposed by Tonon and Vandin (2019) would *not* be an ε-AUS for the null model. Since we want to show an EUS for the null model proposed by Tonon and Vandin (2019), we make the same assumption.

Tonon and Vandin (2019, Sect. III.A) present an ε-AUS for sampling datasets from $\mathcal{Z}$ *almost uniformly*, taking an MCMC approach. The idea is to start from $\mathcal{D}$ and perform a number of *itemsets swaps*, i.e., swap an itemset A from a transaction $t_A \in \mathcal{D}$ with another itemset B from another (or the same) transaction t_B, where A and B are chosen uniformly at random with replacement from the bag of the $m \doteq \sum_{t \in \mathcal{D}} |t|$ itemsets participating in the transactions in $\mathcal{D}$. The resulting Markov chain, where states are datasets in $\mathcal{Z}$ and there is an edge from $\mathcal{D}'$ to $\mathcal{D}''$ iff $\mathcal{D}''$ can

[2] This constraint is not the same as requiring the multi-supports of *all* itemsets be preserved.

be obtained from $\mathcal{D}'$ with a single swap, has a uniform *stationary* distribution over $\mathcal{Z}$. Tonon and Vandin (2019, Thm. 2) show an upper bound $O(m^2 \log {}^m/_\varepsilon)$ to the mixing time $\tau(\varepsilon)$ of the Markov chain, i.e., the number of itemsets swaps needed for the distribution of the obtained dataset to have total-variation distance at most ε from the desired uniform stationary distribution.

We now discuss a different way to look at sampling of datasets from $\mathcal{Z}$. We first use this new point of view to derive a better upper bound to the mixing time $\tau(\varepsilon)$ of the "itemset-swaps" ε-AUS. Afterwards, we present an EUS for this null model, which, in addition to giving "perfect" samples, is much more computationally efficient than the ε-AUS (even with the improved mixing time), both theoretically (see below), and experimentally, as we show in Sect. 5.

The idea is to look at any dataset $\mathcal{D}' \in \mathcal{Z}$ as an m-dimensional vector of itemsets, as follows. Let $t'_1, \ldots, t'_{|\mathcal{D}'|}$ be the transactions in $\mathcal{D}'$ (recall that, in addition to holding $|t'|_i = |t|_i$, $1 \leq i \leq |\mathcal{D}|$, the ordering of these transactions is also fixed, in the null model, to differentiate datasets built over the same set of transactions). We can represent $\mathcal{D}'$ as the vector $\mathsf{v}(\mathcal{D}')$ that has as the first $|t'_1|$ components, the corresponding itemsets from t_1, in the order they appear in this transaction, followed, as the successive $|t'_2|$ components, by the itemsets in t'_2, in the order they appear in this transaction, and so on, until the m-th component is the last itemset participating in $t'_{|\mathcal{D}'|}$. The vector $\mathsf{v}(\mathcal{D})$ of the observed dataset can be computed with a single pass over it, and, given a vector, obtaining the corresponding dataset is equally efficient.

Fact 1 *For any two* distinct $\mathcal{D}', \mathcal{D}'' \in \mathcal{Z}$*, it holds* $\mathsf{v}(\mathcal{D}') \neq \mathsf{v}(\mathcal{D}'')$.

Fact 2 *Let* $\mathcal{D}' \in \mathcal{Z}$ *(possibly* $\mathcal{D}' = \mathcal{D}$*). Then* $\mathsf{v}(\mathcal{D}')$ *is a* permutation *of* $\mathsf{v}(\mathcal{D})$.

Fact 3 *Let* $\mathcal{D}' \in \mathcal{Z}$. Any *permutation of the order of the components of* $\mathsf{v}(\mathcal{D}')$ *gives a vector* w *such that there exists a* $\mathcal{D}'' \in \mathcal{Z}$ *(potentially* $\mathcal{D}'' = \mathcal{D}'$*) for which* $\mathsf{v}(\mathcal{D}'') = \mathsf{w}$.

When there is at least one itemset A *with* $\rho_{\mathcal{D}}(A) > 1$*, there are* many *permutations that map* $\mathsf{v}(\mathcal{D})$ *to* $\mathsf{v}(\mathcal{D}')$.

Facts 1 to 3 imply that the set $\mathcal{S}$ of all $m!$ permutations of the order of the components of $\mathsf{v}(\mathcal{D})$ (from now on, we say "permutations of $\mathsf{v}(\mathcal{D})$" to refer to the permutations of the order of its components) is partitioned into equivalence classes in such a way that there is an equivalence class $C_{\mathcal{D}'}$ for each and only the datasets $\mathcal{D}' \in \mathcal{Z}$ (potentially $\mathcal{D}' = \mathcal{D}$) such that $C_{\mathcal{D}'}$ contains all and only the permutations of $\mathsf{v}(\mathcal{D})$ that result in $\mathcal{D}'$. We now show a key result (Lemma 2) about the sizes of these equivalence classes, which is at the core of our improved mixing time bound for Tonon and Vandin (2019)'s ε-AUS, and of our EUS.

Lemma 2 *All classes have the same size, i.e.,* $|C_{\mathcal{D}}| = |C_{\mathcal{D}'}|$*, for any* $\mathcal{D}' \in \mathcal{Z}$.

Proof Any permutation g in $C_{\mathcal{D}}$ maps $\mathsf{v}(\mathcal{D})$ to itself by only permuting components that have identical itemsets (potentially permuting each component with itself, thus the identity permutation is in $C_{\mathcal{D}}$). That is, if $g(i) = j$, $1 \leq i, j \leq m$, $i \neq j$, then it must be that the itemset at index i of $\mathsf{v}(\mathcal{D})$ is identical to the itemset at index j.

Let $\mathcal{D}' \in \mathcal{Z}$ and fix $g \in C_{\mathcal{D}'}$. Any $r \in C_{\mathcal{D}'}$ can be seen as the composition of g with some $h \in C_{\mathcal{D}}$ (i.e., $r(i) = g(h(i))$, $1 \leq i \leq m$), specifically with h such that $h(i) = g^{-1}(r(i))$, $1 \leq i \leq m$. The permutation h belongs to $C_{\mathcal{D}}$ because, since both g and r belong to $C_{\mathcal{D}'}$, applying g^{-1} (which exists because g is a permutation) to $\mathsf{v}(\mathcal{D}')$ must result in $\mathsf{v}(\mathcal{D})$.

From the above paragraph, it must be $|C_{\mathcal{D}'}| \leq |C_{\mathcal{D}}|$. Conversely, for every different $h \in C_{\mathcal{D}}$, we obtain a different $r \in C_{\mathcal{D}'}$ when we compose g with h. Thus it must also be $|C_{\mathcal{D}'}| \geq |C_{\mathcal{D}}|$, and our proof is complete. □

We then have the following equivalence between drawing datasets from $\mathcal{Z}$ and drawing permutations of $\mathsf{v}(\mathcal{D})$.

Corollary 1 *Let g be a permutation of $\mathsf{v}(\mathcal{D})$ drawn* uniformly at random *from all permutations of $\mathsf{v}(\mathcal{D})$. Then the dataset $\mathcal{D}'$ corresponding to the vector w obtained by applying g to $\mathsf{v}(\mathcal{D})$ is drawn* uniformly at random *from $\mathcal{Z}$.*

This equivalence allows us to give a better analysis of the mixing time of the "itemset-swap" ε-AUS by Tonon and Vandin (2019), by drawing from the rich literature on card shuffling (Levin and Peres 2017, Ch. 8). Indeed the idea of shuffling a deck of cards (i.e., obtaining a uniform random permutation of the set of cards) by repeatedly swapping two cards chosen uniformly at random has been studied deeply. Diaconis and Saloff-Coste (1993, Example 4A) show a $O(m \log m \log \varepsilon^{-1})$ upper bound to the number of swaps needed (see also (Jonasson 2012, Sect. 1)), which is optimal (Wilson 2004), and directly applies to the mixing time of the itemset-swap ε-AUS by Tonon and Vandin (2019), thus improving their $O(m^2 \log {}^{m}/_{\varepsilon})$ bound. This improved bound may also help explaining why, in their experiments, Tonon and Vandin (2019, Sect. IV.C) observed the Markov chain converging to the uniform distribution much faster than explained by their bound.

The restriction that the random permutation of $\mathsf{v}(\mathcal{D})$ can only be obtained by performing itemset swaps starting from $\mathsf{v}(\mathcal{D})$ is *unnecessary*: there is *no reason* to impose such constraint. Drawing a permutation of $\mathsf{v}(\mathcal{D})$ uniformly at random can be done using the Fisher-Yates shuffle (Knuth 1998, Sect. 3.4.2) with input $\mathsf{v}(\mathcal{D})$. The output will be a vector w that is obtained by applying a uniformly-chosen permutation to $\mathsf{v}(\mathcal{D})$. The Fisher-Yates shuffle runs in time $O(m)$. Parallel algorithms outputting such a w are also available (Bacher et al. 2015). The dataset $\mathcal{D}'$ corresponding to w is the dataset whose first transaction t'_1 contains exactly, in order, the first $|t_1|$ components of w, in the same order as they appear in w, and whose second transaction t'_2 contains exactly, in order, the successive $|t_2|$ components, and so on. In conclusion, we have a EUS for this first null model, which is more efficient than the existing ε-AUS in both probabilistic and computational terms, and thus should be preferred. In Sect. 5 we show that our EUS is faster also in practice.

4.2.2 Null model #2: fixed multi-supports, and transaction lengths and item-lengths

We now introduce a first novel null model. As in Sect. 4.2.1, we fix an arbitrary ordering of the transactions $t_1, \ldots, t_{|\mathcal{D}|}$ of the observed dataset $\mathcal{D}$, and will consider two datasets to be different even if they only differ by the ordering of their transactions. The null set $\mathcal{Z}$ contains all and only the datasets $\mathcal{D}'$ with $|\mathcal{D}|$ transactions such that, *in addition* to meeting the constraints from null model #1 (Sect. 4.2.1), also satisfy the condition that the transactions $t'_1, \ldots, t'_{|\mathcal{D}|}$ of $\mathcal{D}'$ have $\|t'_i\| = \|t_i\|$, $1 \leq i \leq |\mathcal{D}|$, i.e., the *item-lengths* of the transactions of $\mathcal{D}$ are preserved. The null distribution π is the uniform distribution over $\mathcal{Z}$.

We now give an EUS for this null model. Consider the m itemsets in $\mathcal{D}$, and let ℓ be the length of the longest one. Consider the ℓ vectors $\mathsf{z}_1, \ldots, \mathsf{z}_\ell$, each z_i containing

all and only the itemsets of length i appearing in $\mathcal{D}$, *in the same order* as they appear in the vector $\mathsf{v}(\mathcal{D})$ as defined in Sect. 4.2.2. Let z_i be the number of components of z_i, $1 \leq i \leq \ell$ (the sum of the z_i's is m). Let p_i, $1 \leq i \leq \ell$, be a vector of z_i components, whose j^{th} component is the *index* of the component of $\mathsf{v}(\mathcal{D})$ where the itemsets $\mathsf{z}_i[j]$ appears, $1 \leq j \leq z_i$. All these vectors can be computed in a single pass over the dataset, and can be re-used by the procedure for multiple samples. Our method first permutes, uniformly at random, each vector z_i, $1 \leq i \leq \ell$, then creates a vector w of m components by iterating over each p_i and setting the component at index $\mathsf{p}_i[j]$ of w to the itemset $\mathsf{z}_i[j]$ (i.e., the itemset in component j of the permuted z_i), $1 \leq j \leq z_i$. A first "temporary" dataset T is obtained from the vector w, as we did in Sect. 4.2.1. The order of the itemsets in each transaction in T is then permuted uniformly at random to obtain the output dataset $\mathcal{D}''$. This permutation is necessary to ensure that *only* the desired constraints are satisfied.[3]

Let us now show that this procedure is an EUS for the null model. Consider the set $\mathcal{S} \doteq S_1 \times \cdots \times S_\ell$, where each $\mathcal{S}_i$ is the set of all $z_i!$ permutations of z_i, $1 \leq i \leq \ell$, and the set $\mathcal{Q} \doteq Q_1 \times \cdots \times Q_{|\mathcal{D}|}$, were each Q_i is the set of all $|t_i|!$ permutations of transaction t_i, $1 \leq i \leq |\mathcal{D}|$. Our algorithm can be seen as choosing an element $\mathsf{r} \doteq ((r_1, \ldots, r_\ell), (q_1, \ldots, q_{|\mathcal{D}|}))$ from $\mathcal{S} \times \mathcal{Q}$ uniformly at random, by choosing each permutation r_i uniformly at random from S_i, $1 \leq i \leq \ell$, and each permutation q_i from Q_i $1 \leq i \leq |\mathcal{D}|$. The set $\mathcal{S} \times \mathcal{Q}$ is therefore partitioned into equivalence classes, one and only one class $C_{\mathcal{D}'}$ for each dataset $\mathcal{D}' \in \mathcal{Z}$, containing all and only the elements of $\mathcal{S} \times \mathcal{Q}$ that, when the r_i's are applied to the vectors z_i, $1 \leq i \leq \ell$, and then the q_i's are applied to the resulting transactions of the "temporary' dataset T, make our procedure output the dataset $\mathcal{D}'$.

Lemma 3 *The classes have all the same size, i.e., $|C_{\mathcal{D}'}| = |C_{\mathcal{D}}|$, for every $\mathcal{D}' \in \mathcal{Z}$.*

The proof follows the steps similar to the one for Lemma 2, but taking into account the different nature of the elements in the classes (vectors of permutations in this case, not single permutations).

Corollary 2 *The procedure is an EUS for the null model.*

One can easily obtain an ε-AUS for this null model by repeatedly swapping itemsets of the same length: first a vector z_i is chosen with a probability proportional to z_i, then two itemsets, chosen uniformly at random with replacement among those in z_i are swapped. We use this ε-AUS as a baseline for the experimental evaluation of our EUS in Sect. 5.

4.2.3 Null model #3: fixed itemset supports and multi-supports, and fixed transaction lengths

In this null model, we still assume a fixed ordering of the transactions $t_1, \ldots, t_{|\mathcal{D}|}$ of the observed dataset $\mathcal{D}$, but, *differently from the previous models*, we will *not* consider two datasets different when they only differ by the ordering of transactions of the same length (see below for how to reintroduce this constraint, if so desired).

[3] One can avoid this step and obtain a uniform sample from a null model that, in addition to all the constraints we defined, also preserves the *order of the lengths* of the itemsets in the transactions of $\mathcal{D}$.

E.g., the two datasets from (4) are the *same* dataset in this null model. The null set $\mathcal{Z}$ contains all and only the datasets $\mathcal{D}'$ with $|\mathcal{D}|$ transactions that, in addition to satisfying the constraints from Sect. 4.2.1, are also such that, for every itemset A participating in at least one transaction t, it holds that $\sigma_{\mathcal{D}'}(A) = \sigma_{\mathcal{D}}(A)$, i.e., the supports of itemsets participating in the transactions are preserved.[4] Recall that $\sigma_{\mathcal{D}'}(A)$ is the number of transactions in which A participates, with no consideration as to whether a transaction contains multiple copies of A. The difference between null model #1 (Sect. 4.2.1) and this one is that the former only preserves the *multi-support*, while this one also preserves the support. We give an ε-AUS for this null model, based on repeated itemset swaps as follows.

Consider the vector $\mathsf{v}(\mathcal{D})$ corresponding to $\mathcal{D}$. Starting from $\mathsf{w} = \mathsf{v}(\mathcal{D})$, our sampling procedure repeatedly selects a pair (i, j) where i and j are drawn uniformly at random, with replacement, from $[1, m]$, and obtains a new vector w' by swapping the itemsets I and J in the i^{th} and j^{th} component of w, *iff* the resulting vector w' is such that the supports of I and J in the dataset corresponding to w' are the same as in $\mathcal{D}$. This condition is easy to check: let $\mathcal{D}_{\mathsf{w}}$ be the dataset corresponding to w, and let t_I (resp. t_J) be the transaction of $\mathcal{D}_{\mathsf{w}}$ that contains I (resp. J) at the i^{th} (resp. j^{th}) component of w. Swapping I and J is allowed iff the following condition holds:[5]

$$\begin{aligned}
&((t_I \text{ contains a single copy of } I \wedge t_J \text{ contains no copy of } I)\\
&\quad \vee \text{ both } t_I \text{ and } t_J \text{ contain multiple copies of } I)\\
\wedge&((t_J \text{ contains a single copy of } J \wedge t_I \text{ contains no copy of } J)\\
&\quad \vee \text{ both } t_J \text{ and } t_I \text{ contain multiple copies of } J) \ .
\end{aligned}$$

If this condition is not satisfied, then no swap is performed, and $\mathsf{w}' = \mathsf{w}$. Then the procedure sets $\mathsf{w} = \mathsf{w}'$ and iterates. After $O(m^2 \log {}^{m}\!/_{\varepsilon})$ iterations (see Lemma 5), the dataset $\mathcal{D}'$ corresponding to the last w' is returned in output as the sample.

We now move to show that this procedure is a (fully-polynomial) ε-AUS. Consider the Markov chain C whose set of states is $\mathcal{Z}$ and whose transition probabilities arise from the procedure. We first show that it has a uniform stationary distribution (Lemma 4), and then show an upper bound to its mixing time $\tau(\varepsilon)$ (Lemma 5).

Lemma 4 *The Markov chain C has a unique uniform stationary distribution.*

Proof The Markov chain C is clearly aperiodic, because drawing a pair (i, i) keeps the chain in the same state. It is also irreducible, as it is possible go from any dataset $\mathcal{D}' \in \mathcal{Z}$ to any other dataset $\mathcal{D}'' \in \mathcal{Z}$ with a sequence of itemset swaps as above. Thus C has a unique stationary distribution.

For any ordered pair $(\mathcal{D}', \mathcal{D}'')$ of not-necessarily distinct states, let (a, b) be any pair in $\{(i, j) \ : \ 1 \leq i, j \leq m\}$ such that the corresponding itemset swap moves the chain from $\mathcal{D}'$ to $\mathcal{D}''$, if such a pair (a, b) exists. If it exists, the same pair would move the chain from $\mathcal{D}''$ to $\mathcal{D}'$. In this case, the transition probability from $\mathcal{D}'$ to $\mathcal{D}''$ must be the same as the one from $\mathcal{D}''$ to $\mathcal{D}'$. If (a, b) does not exist, then these transition probabilities must both be zero. Therefore, the transition matrix of C is clearly symmetrical. A symmetrical transition matrix is doubly-stochastic, and a Markov chain with a doubly-stochastic transition matrix has the uniform stationary distribution (Motwani and Raghavan 1995, Probl. 6.6).

[4] This constraint is not the same as requiring that the supports of *all* itemsets are preserved.

[5] We denote with "$\wedge$" the logical AND, and with "$\vee$" the logical OR.

Lemma 5 *Let $\varepsilon \in [0,1]$. For the Markov chain C, it holds $\tau(\varepsilon) = O(m^2 \log {}^{m}/_{\varepsilon})$.*

The proof uses the path coupling technique (Levin and Peres 2017, Ch. 14) and follows steps similar to that of (Tonon and Vandin 2019, Thm. 2).[6]

Proof Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two datasets whose corresponding vectors $\mathsf{v}(\mathcal{D}_1)$ and $\mathsf{v}(\mathcal{D}_2)$ differ in exactly two components, w.l.o.g. those at indices a and b, $1 \leq a \leq b \leq m$. We define the distance $\mathsf{q}(\mathcal{D}', \mathcal{D}'')$ between two datasets (i.e., states of the chain) as the number of components of corresponding vectors in which they differ, thus $\mathsf{q}(\mathcal{D}_1, \mathcal{D}_2) = 2$. It must be that the itemsets at indices a and b can be swapped, i.e., there is a non-zero probability of moving along the Markov chain from $\mathcal{D}_1$ to $\mathcal{D}_2$ and vice-versa. Consider now the following coupling where the first Markov chain C_1 is C and we assume it to be in state $\mathcal{D}_1$, and the second Markov chain C_2 is assumed to be in state $\mathcal{D}_2$ and its transitions are defined on the basis of the transitions of C_1 as follows. Suppose that C_1 samples the indices (x, y) from $\{(i, j) \ : \ 1 \leq i, j \leq m\}$. Let $\mathcal{D}'_1$ be the state that C_1 moves to after performing the action corresponding to (x, y). The state $\mathcal{D}'_2$ to which C_2 moves from $\mathcal{D}_2$ is defined as follows: *(i)* if $x = a$ and $y = b$, then $\mathcal{D}'_2 = \mathcal{D}_2$; *(ii)* if $x = b$ and $y = a$, then $\mathcal{D}'_2 = \mathcal{D}_2$; *(iii)* if $x = y = a$ or $x = y = b$ then $\mathcal{D}'_2 = \mathcal{D}_1$; *(iv)* otherwise the chain follows the same transition that the original chain would have performed from D_2 when sampling the pair (x, y). In cases *(i)*, *(ii)*, and *(iii)*, the distance $\mathsf{q}(\mathcal{D}'_1, \mathcal{D}'_2)$ is zero, while in case *(iv)* it could either go stay at two, grow to four, or go to zero depending on whether the action corresponding to (x, y) can be performed by neither of C_1 and C_2, both of them, or only one of them (not respectively). The probability of any transition is the same in both chains, and in particular the probability of being in one of the first three cases is ${}^{4}/_{m^2}$. Thus the expectation of the distance between the next states of C_1 and C_2, conditioned on them currently being in states at distance two, is at most $4(1 - {}^{4}/_{m^2})$. We obtain the lemma through standard path coupling results (Levin and Peres 2017, Coroll. 14.8), using the fact that the maximum distance between two states is m. □

From Lemmas 4 and 5 we can conclude that our procedure is a (fully-polynomial) ε-AUS for the null model we introduced.

Reintroducing the "different-ordering-different-datasets" constraint It may be desirable that datasets that only differ for the ordering of transactions of the same length be different, e.g., that the two datasets from (4) be different, as was assumed in null models #1 and #2. In this case, it is possible to obtain an ε-AUS by taking the output $\mathcal{D}''$ of the procedure described above and permuting the *order* of transactions of the same length (which is not the same as permuting the itemsets in each transaction), uniformly at random among all possible orderings.

5 Experiments

We present here the results of our experimental evaluation of SPEck, instantiated with the sampling procedures described in Sect. 4.2.

[6] We conjecture that it should be possible to prove $\tau(\varepsilon) = O(m \log m \log \varepsilon^{-1})$.

Dataset	$\lvert D\rvert$	$\lvert\mathcal{A}\rvert$	avg. $\lVert t\rVert$	repeated items in t	θ	$\lvert\mathsf{F}_{\mathcal{D}}(\theta)\rvert$
Bible	36369	13905	21.6	yes	0.1	174
Bike	21078	67	7.28	yes	0.025	163
Fifa	20450	2990	36.2	yes	0.275	182
Leviathan	5835	9025	33.8	yes	0.15	225
Sign	730	267	52.0	no	0.4	518

Table 1 Dataset properties, min. frequency threshold θ, and number of frequent sequences.

Goals EUS's are always to be preferred to ε-AUS's from a probabilistic point of view. Thus, we focus on evaluating whether our EUS's are *faster* than ε-AUS's for the same null model, by comparing them on the basis of the time needed to output a single sample, and when used in SPEck. We also evaluate the *scalability* of the procedure by measuring how the sampling time changes as a function of the number of itemsets in the dataset (i.e., the quantity m). Finally, we evaluate the effects of the parameter θ on the runtime of SPEck.

Implementation and environment We implement SPEck in Java 8, by modifying the publicly-available implementation of ProMiSe (Tonon and Vandin 2019), which is based on the Apache Spark framework.[7] SPEck, like ProMiSe, is embarrassingly parallel: the work on each sampled dataset can proceed independently from, thus in parallel with, the work done on the other sampled datasets, with a final reduction that uses the results from each sampled dataset to compute the collection of significant sequences. We performed the experiments on an x86–64 machine with 2 Intel® Xeon® 4210R CPUs (40 threads in total), 348GB of RAM, running FreeBSD 14.

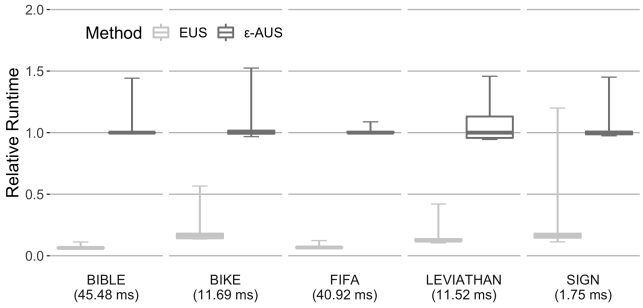
Datasets and parameters We use five real datasets, all publicly available:[8]

- Bible: a conversion of the bible. Each word is an itemset of length one and each sentence is a transaction.
- Bike: data from Los Angeles Metro Bike Share. Each item is a bike station and a transaction is the sequence of stations where a bike has been.
- Fifa: click-stream data from the website of the 1998 Fifa World Cup. An item represents a unique web page.
- Leviathan: a conversion of T. Hobbes's *Leviathan*. Each word is an itemset of length one and each sentence is a transaction.
- Sign: a dataset of sign language utterance.

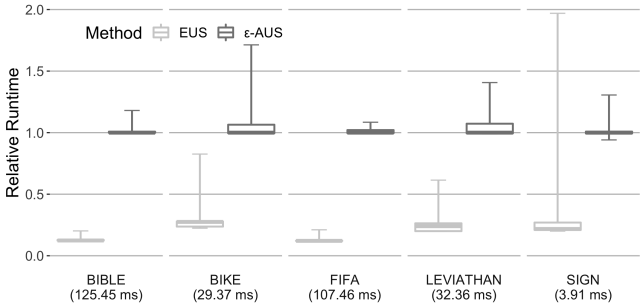
The main dataset statistics are shown in Table 1, together with the minimum frequency threshold θ we used for each dataset and the size of the collection of frequent sequences w.r.t. θ. When running SPEck, we use $T = 10000$ (which is a multiple of 40, the number of processors we use) and $P = 100$. These numbers are in line with those used by Tonon and Vandin (2019) for a similar set of experiments, and we verified empirically that using larger values for them would have a negligible effect on the approximations of the p-values and the adjusted critical values. There is no downside to using even larger values, apart from a longer running time, which would be partially offset by a higher level of parallelism. It is an interesting direction for

[7] Our code, including the data and scripts to reproduce all our results and figures, is available from `https://github.com/acdmammoths/SPEck-code/`.

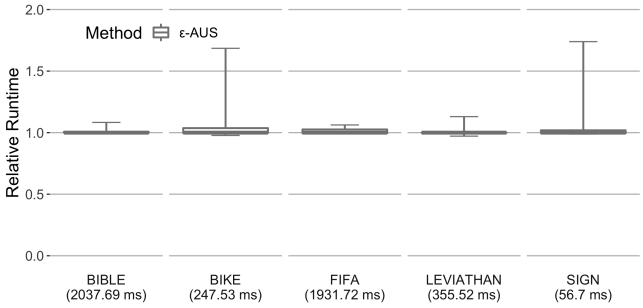
[8] `https://github.com/VandinLab/PROMISE/tree/master/data`.



(a) Null model #1



(b) Null model #2



(c) Null model #3

Fig. 1 Distribution of the relative runtimes on real datasets, normalized by the median, over 100 runs, of the runtime for the ε-AUS (which therefore corresponds to the 1.0 line; the absolute runtimes for this median are shown under the dataset name). The whiskers corresponds to minimum and maximum, the extremes of the box to 1^{st} and 3^{rd} quartile, and the line crossing the box to the median.

future work to incorporate the error in the approximation of the p-values and in the FWER due to the use of finite values of T and P, in order to make the process of mining significant patterns even more statistically rigorous. We fix the acceptable FWER δ to 0.05. When using an ε-AUS to generate the samples, we use $2m$ as the number of swaps to perform before taking each sample, which we checked experimentally to be sufficient for the state distribution to be extremely close to uniform, by

looking at the convergence of the average relative difference between the frequency of a frequent sequence in the observed dataset and the frequency in a random dataset, in the same way as done by Tonon and Vandin (2019, Fig. 2).

To evaluate the scalability of the sampling procedures, we use artificial datasets with $|\mathcal{D}| \in \{1000, 3162, 10000, 31623, 10000\}$, generated using the IBM Quest Dataset Generator[9] (Agrawal and Srikant 1994), run with the default parameters.

To evaluate the effect of the minimum frequency threshold θ on the runtime and on the number of significant frequent patterns we use the publicly available BIBLE dataset, with $\theta \in \{0.031, 0.043, 0.06, 0.1\}$. The choice of these values is guided by the number of frequent patterns w.r.t. θ (see Table 3, third column from the left).

		Runtime on Dataset (s)				
Null Model	Method	Bible	Bike	Fifa	Leviathan	Sign
#1	EUS	2454	343	2308	432	90
	ε-AUS	4139	432	3341	582	72
#2	EUS	3386	552	3447	774	128
	ε-AUS	8494	1268	5702	1132	157
#3	ε-AUS	4950	559	7249	1594	272

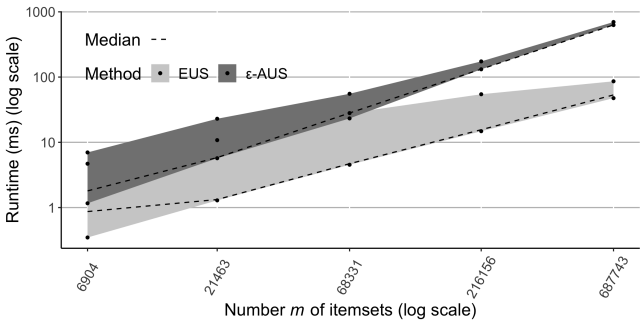
Table 2 Runtime of SPEck with different sampling procedures. EUS's, when available, are much faster than ε-AUS's, in addition to be preferrable from a probabilistic point of view.

Results on real datasets In Fig. 1 (one figure per null model) we report the *relative* runtimes, on the five real datasets, of the EUS we introduce for that model (or ε-AUS, in the case of Model #3 in Fig. 1c), and a baseline ε-AUS, when available (for Model #1 it is Tonon and Vandin (2019)'s procedure; for Model #2 it is the "same-length-itemset-swap" procedure described at the end of Sect. 4.2.1). The box-and-whiskers plots show the minimum, first quartile, median, third quartile, and maximum of the runtimes. The runtimes are *normalized* by the *median*, over 100 runs, of the ε-AUS's runtime, so, e.g., the median of the ε-AUS's runtimes is always on the 1.0 line. Absolute runtimes for the median of the ε-AUS (i.e., the normalization factors) are reported in parentheses under each dataset's name.

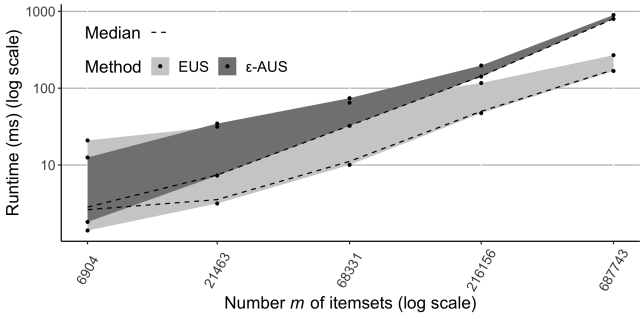
For Models #1 and #2, it is evident that the EUS's are much faster than the ε-AUS's, which confirms that *our EUS's should be preferred on both probabilistic and computational grounds*. A comparison of the median runtimes reveals speed-up factors between 5.08 (for Sign, on Model #2) and 26.7 (for FIFA, on Model #1). We can also appreciate from the figures that, in general, the runtime of EUS's has lower variance than the runtime of ε-AUS's. The exception in this case is Sign, which is a relatively small dataset (see Table 1)–even the ε-AUS takes only a few milliseconds to sample a dataset from the null models when given this dataset as input, so a larger variance in the runtimes is to be expected, and is mostly due to outliers (see also the next experiment).

In Table 2 we report, for each null model and each real dataset, the running time of SPEck using EUS's and ε-AUS's. The reduction in runtime when using exact sampling procedures over approximate ones is significant. The reason the speedups

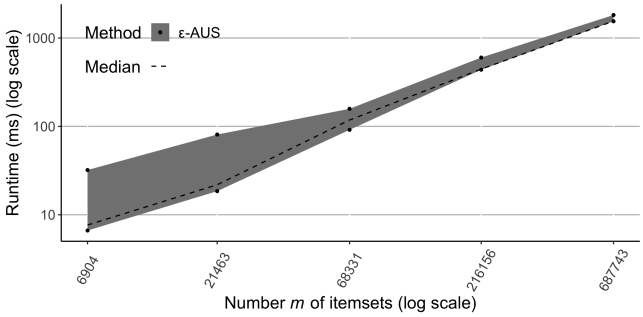
[9] Available from `https://github.com/acdmammoths/datasetgenerator`.



(a) Null model #1



(b) Null model #2



(c) Null model #3

Fig. 2 Absolute runtimes on artificial datasets as function of the total number m of itemsets. The median is over 100 runs and the shaded area goes from the minimum to the maximum runtime. The EUS's scale as well as the ε-AUS's.

are not the same as when evaluating the runtimes of the sampling procedures in isolation (as we did above), is that a large portion of SPEck's runtime is spent on operations other than sampling datasets (e.g., computing the frequent sequences of the sampled datasets), which are not impacted by the choice of sampling procedure. We remark that it is not very interesting to compare the runtimes *across* different null models: the meaning of *significant pattern* changes with the model, and the choice of model should be *very deliberate* on the user's side, depending on the desired meaning.

Thus it should not be surprising that the sampling procedures are different and take different amounts of time.

Scalability Figure 2 compare how well the EUS's and ε-AUS's scale as the total number m of itemsets grows. We used our artificial datasets in this experiment, and repeated it 100 times. The figures show the absolute runtimes of the various sampling methods. The shaded areas go from the minimum to the maximum runtime over 100 runs, and we also report the median (dashed lines). We can see how both EUS's and ε-AUS's scale in a similar way, with EUS's remaining much faster. We can appreciate again how the variance in the runtimes decreases as m grows. This experiment confirms that the EUS's we present should be preferred over ε-AUS's.

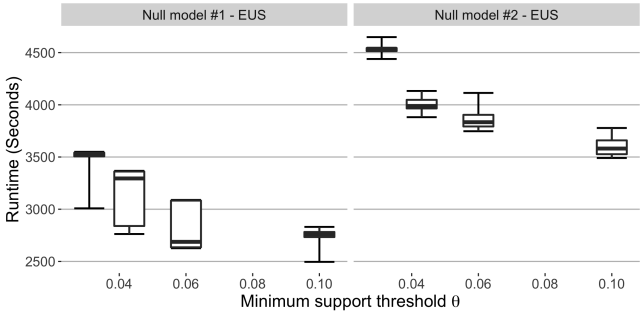


Fig. 3 Runtimes of SPEck on BIBLE for $\theta \in \{0.031, 0.043, 0.06, 0.1\}$ (see text for rationale).

Impact of the minimum frequency threshold We used $\theta \in \{0.031, 0.043, 0.06, 0.1\}$ to study its impact on the runtime and on the number of significant frequent patterns. The choice of values is guided by the number of frequent patterns in BIBLE w.r.t. θ (third column from the left in Table 3).

Figure 3 shows that as the value of θ increases, the runtime of SPEck decreases. This behavior is expected, as more time is required to mine a dataset at a lower threshold. Indeed the behavior is essentially linear with the number of frequent patterns. The runtime variance is explained with the next set of results.

			SFSPs in iteration #				
Method	θ	$\lvert\mathsf{F}_{\mathcal{D}}(\theta)\rvert$	1	2	3	4	5
Null model #1 — EUS	0.031	2074	0	0	0	0	0
	0.043	1035	786	0	787	0	787
	0.06	507	366	0	0	366	0
	0.1	174	119	120	120	121	119
Null model #2 — EUS	0.031	2074	0	0	0	0	0
	0.043	1035	0	787	0	788	787
	0.06	507	367	366	367	365	0
	0.1	174	119	120	120	119	121

Table 3 No. of SFSPs in BIBLE returned by SPEck for different values of θ over five iterations.

We also studied the number of Significant Frequent Sequential Patterns (SFSPs) in BIBLE output by SPECK for different values of θ. We performed each experiment five times, and report the results in Table 3 (we remind the reader, once again, that results for different null models should not be compared to each other). For relatively small values of θ, SPECK reported, in some runs, no significant patterns. We remark that outputting no significant patterns is perfectly fine and in line with the guarantees on the FWER,[10] and it is just a sign of low statistical power. As discussed in Sect. 3.1, we use the Westfall-Young method to control the FWER, which provides more statistical power than the Bonferroni correction would. However, when θ is small, the set of hypotheses that even the Westfall-Young method has to consider is large, because it is related to the number of patterns in the observed dataset that have a frequency (in the observed dataset) lower-than-but-close-to θ, and this number increases with θ. Any A of these patterns may have, in one $\mathcal{D}_i$ of the P sampled datasets used to compute the adjusted critical value α_{WY} (see (3)), a frequency not smaller than θ, but A may not have such a frequency in any of the T datasets used to estimate A's p-value $\tilde{p}_{\mathcal{D}_i}(A)$ in $\mathcal{D}_i$ (see (1)). This fact in turn leads to the lowest possible minimum p-value $\check{p}_i = 1/(T+1)$ for $\mathcal{D}_i$ (see (2)), and thus to a low adjusted critical value $\alpha_{\text{WY}} = 1/(T+1)$, when there are such A's in a fraction at least δ of the P datasets. Since this value is the minimum possible (empirical) p-value that a pattern may have, no pattern can actually be marked as significant by SPECK. In other words, SPECK's statistical power decreases with θ.

We remark that this behavior is not specific of SPECK, but it affects any method that uses an empirical estimate of the p-values, and an empirical estimate of the minimum p-value distribution, which make the p-values and the adjusted critical value only take discrete values. On one hand, this issue cannot be avoided, because we are forced to use empirical estimations for these quantities and distributions, as they cannot be derived exactly, on the other hand, intuitively, this issue could be mitigated by increasing T and P. As our results for $\theta = 0.06$ and $\theta = 0.043$ show, SPECK does not always report zero significant patterns: it really depends on the datasets sampled in each iteration. Developing methods that *(1)* have high statistical power in all situations; and *(2)* control the FWER, is the "holy grail" of researchers in this field, and a natural (if challenging) direction for future work.

6 Conclusion

We presented SPECK, a framework for mining statistically-significant sequential patterns from large datasets under different null models, using the Westfall-Young resampling approach. We study a null model first proposed by Tonon and Vandin (2019) and introduce two novel null models that preserve different properties of the observed dataset. Our main algorithmic contributions are new methods to sample datasets from the null model. For the previously-studied model, we give an *Exact* Uniform Sampler (EUS) that greatly improves, both in probabilistic and computational terms, over the existing ε-*Almost* Uniform Sampler (ε-AUS), for which we also present an improved analysis of the mixing time. For the novel models we introduce, we give EUS's and/or ε-AUS's. The results of our experimental evaluation show that our EUS's are much faster than ε-AUS's, and thus they should be preferred.

[10] The easiest way for an algorithm to offer guarantees on the FWER is to never report anything as significant. It would not be a very useful algorithm though.

Acknowledgements We thank Andrea Tonon and Fabio Vandin for helpful clarifications on the settings of statistically-significant sequential pattern mining, and Nathan Pflueger for a fruitful discussion leading to the proof of Lemma 2. This work is supported in part by NSF award IIS-2006765.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proc. 20th Int. Conf. Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB '94, pp 487–499

Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering„ IEEE, ICDE'95, pp 3–14

Bacher A, Bodini O, Hollender A, Lumbroso J (2015) Mergeshuffle: A very fast, parallel random permutation algorithm. arXiv preprint arXiv:150803167

Bonferroni CE (1936) Teoria statistica delle classi e calcolo delle probabilità. Pubblicazioni del Regio Istituto Superiore di Scienze Economiche e Commerciali di Firenze 8:3–62

Diaconis P, Saloff-Coste L (1993) Comparison techniques for random walk on finite groups. The Annals of Probability 21(4):2131–2156

Duivesteijn W, Knobbe A (2011) Exploiting false discoveries–statistical validation of patterns and quality measures in subgroup discovery. In: 2011 IEEE 11th International Conference on Data Mining, IEEE, pp 151–160

Feremans L, Cule B, Goethals B (2018) Mining top-k quantile-based cohesive sequential patterns. In: Proceedings of the 2018 SIAM international conference on data mining, SIAM, pp 90–98

Fournier-Viger P, Wu CW, Gomariz A, Tseng VS (2014) VMSP: Efficient vertical mining of maximal sequential patterns. In: Canadian conference on artificial intelligence, Springer, pp 83–94

Fumarola F, Lanotte PF, Ceci M, Malerba D (2016) CloFAST: closed sequential pattern mining using sparse and vertical id-lists. Knowledge and Information Systems 48(2):429–463

Gionis A, Mannila H, Mielikäinen T, Tsaparas P (2007) Assessing data mining results via swap randomization. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(3):14

Gwadera R, Crestani F (2010) Ranking sequential patterns with respect to significance. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, pp 286–299

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

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

Hrovat G, Fister I Jr, Yermak K, Stiglic G, Fister I (2015) Interestingness measure for mining sequential patterns in sports. Journal of Intelligent & Fuzzy Systems 29(5):1981–1994

Jonasson J (2012) Mixing times for the interchange process. arXiv preprint arXiv:12106916

Knuth DE (1998) Seminumerical algorithms, The Art of Computer Programming., vol 2, 3rd edn. Addison–Wesley

Lam HT, Mörchen F, Fradkin D, Calders T (2014) Mining compressing sequential patterns. Statistical Analysis and Data Mining: The ASA Data Science Journal 7(1):34–52

Levin DA, Peres Y (2017) Markov chains and mixing times, 2nd edn. American Mathematical Soc.

Low-Kam C, Raïssi C, Kaytoue M, Pei J (2013) Mining statistically significant sequential patterns. In: 2013 IEEE 13th International Conference on Data Mining, IEEE, pp 488–497

Méger N, Rigotti C, Pothier C (2015) Swap randomization of bases of sequences for mining satellite image times series. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp 190–205

Mitzenmacher M, Upfal E (2005) Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press

Mollenhauer D, Atzmueller M (2020) Sequential exceptional pattern discovery using pattern-growth: An extensible framework for interpretable machine learning on sequential data. In: First International Workshop on Explainable and Interpretable Machine Learning (XI-ML)

Motwani R, Raghavan P (1995) Randomized Algorithms. Cambridge University Press

Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu MC (2004) Mining sequential patterns by pattern-growth: The PrefixSpan approach. IEEE Transactions on knowledge and data engineering 16(11):1424–1440

Pellegrina L, Vandin F (2020) Efficient mining of the most significant patterns with permutation testing. Data Mining and Knowledge Discovery 34:1201–1234

Pellegrina L, Riondato M, Vandin F (2019a) Hypothesis testing and statistically-sound pattern mining. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, New York, NY, USA, KDD '19, pp 3215–3216, DOI 10.1145/3292500.3332286, URL http://doi.acm.org/10.1145/3292500.3332286

Pellegrina L, Riondato M, Vandin F (2019b) SPuManTE: Significant pattern mining with unconditional testing. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, New York, NY, USA, KDD '19, pp 1528–1538, DOI 10.1145/3292500.3330978, URL http://doi.acm.org/10.1145/3292500.3330978

Petitjean F, Li T, Tatti N, Webb GI (2016) Skopus: Mining top-k sequential patterns under leverage. Data Min Knowl Discov 30(5):1086–1111

Pinxteren S, Calders T (2021) Efficient permutation testing for significant sequential patterns. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM), SIAM, pp 19–27

Raïssi C, Poncelet P (2007) Sampling for sequential pattern mining: From static databases to data streams. In: Seventh IEEE International Conference on Data Mining, IEEE, ICDM '07, pp 631–636

Raïssi C, Calders T, Poncelet P (2008) Mining conjunctive sequential patterns. Data Mining and Knowledge Discovery 17(1):77–93

Servan-Schreiber S, Riondato M, Zgraggen E (2020) ProSecCo: Progressive sequence mining with convergence guarantees. Knowledge and Information Systems 62(4):1313–1340

Sugiyama M, Llinares-López F, Kasenburg N, Borgwardt KM (2015) Significant subgraph mining with multiple testing correction. In: Proceedings of the 2015 SIAM International Conference on Data Mining, SIAM, pp 37–45

Tatti N (2015) Ranking episodes using a partition model. Data mining and knowledge discovery 29(5):1312–1342

Tonon A, Vandin F (2019) Permutation strategies for mining significant sequential patterns. In: 2019 IEEE International Conference on Data Mining (ICDM), IEEE, pp 1330–1335, full version at https://www.dei.unipd.it/~vandinfa/ICDM19full.pdf

Truong-Chi T, Fournier-Viger P (2019) A survey of high utility sequential pattern mining. In: High-Utility Pattern Mining, Springer, pp 97–129

Westfall PH, Young SS (1993) Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment. Wiley-Interscience

Wilson DB (2004) Mixing times of lozenge tiling and card shuffling Markov chains. The Annals of Applied Probability 14(1):274–325