Using permutation tests to identify statistically sound and nonredundant sequential patterns in educational event sequences

Yingbin Zhang [a*], Luc Paquette [b], Nigel Bosch [c]

[a] Institute of Artificial Intelligence in Education, South China Normal University

[b] Department of Curriculum & Instruction, University of Illinois at Urbana-Champaign

[c] School of Information Sciences and Department of Educational Psychology, University of Illinois at Urbana-Champaign

***Corresponding author**

Yingbin Zhang, Research Fellow

Institute of Artificial Intelligence in Education, South China Normal University

No.55 West of Zhongshan Avenue, Guangzhou, Guangdong 510631, China.

zyingbin@m.scnu.edu.cn

Authors' Note

The R scripts for the permutation tests and simulations are available via the following link: https://github.com/yingbinz/permutation_spm.

Acknowledgment

We thank Prof. Carolyn Anderson for her comments on the early version of the manuscript.

Funding

The author(s) disclosed receipt of the following financial support for the research and/or authorship of this article: This work was supported by the National Key Research and Development Program of China (2023YFC3305704).

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Abstract: Frequent sequential pattern mining is a valuable technique for capturing the relative arrangement of learning events, but current algorithms often return excessive learning event patterns, many of which may be noise or redundant. These issues exacerbate researchers' burden when interpreting the patterns to derive actionable insights into learning processes. This study proposed permutation tests for identifying sequential patterns whose occurrences are statistically significantly greater than the chance value and different from their superpatterns. Simulations demonstrated that the test for detecting sound patterns had a low false discovery rate and high power, while the test for detecting nonredundant patterns also showed a high accuracy. Empirical data analyses found that the patterns detected in training data were generalizable to test data.
Keywords: sequential pattern mining, permutation test, event sequence, behavioral pattern

1 Introduction

Learning is the acquisition process of knowledge and skills (Molenaar, 2014). Behavioral changes resulting from newly acquired knowledge and skills take time to become evident (Soderstrom & Bjork, 2015). Thus, temporality is innate in learning, and an increasing number of researchers have advocated for temporal analysis of learning (Knight et al., 2017; Molenaar, 2014; Reimann, 2009). Molenaar and Wise (2022) delineated four distinctive values of temporal analysis in education: detecting temporal associations between learning events, identifying variation in learning processes, explaining variation in learning outcomes, and boosting the emergence of new questions.

Sequential analyses, a subset of temporal analysis, are a powerful tool for capturing the temporal order of multiple events (Bakeman & Gottman, 1997). Examples are the patterns of ordered events that frequently occur during learning (e.g., *attempt a question → failure → asking for a hint*) and the sequential associations between events that are significant for the learning process (e.g., the transitional probability from an incorrect attempt on a question to asking for a hint). This type of analysis technique can be applied to data that are sequences, which are composed of ordered events, such as action logs in a learning management system, coded thinking aloud data while solving a problem, and coded discourse during collaborative learning. For instance, educational researchers have used lag-sequential analyses (LSA) to reveal the interactions between learners within collaborative learning (Hopcan et al., 2022), *n*-grams to discover differences between incorrect and correct problem-solving behavioral patterns (Ulitzsch et al., 2022), and frequent sequential pattern mining (FSPM) to capture differences in video lecture viewing behaviors between learners with different responses to prompts (Wong et al., 2019).

LSA uses a log-linear model to fit the matrix of event transition frequencies and outputs the statistical significance of event transitions (Bakeman & Gottman, 1997). *N*-grams are a natural language processing tool that represents documents by patterns of consecutive words (Robertson & Willett, 1998). Events in learning processes can be analogous to words. *n*-grams and FSPM can handle longer event patterns than LSA because LSA focuses on event dyads. *n*-grams require the events of a pattern occur consecutively in sequences. This constraint is sensible in natural language processing, as omitting a single word can significantly alter the meaning of a sentence or phrase.

However, for event sequences, the omission of one event may have a less pronounced impact. In this regard, FSPM is more flexible than *n*-grams in the broad fields of educational data mining and learning analytics. FSPM has been used to discover behavioral differences between learners (e.g., high achievement versus low achievement learners), evaluate the effect of intervention on the learning process, extract learning behavioral features for predictive modeling, and mining learning resource accessing patterns for developing recommender systems (Zhang & Paquette, 2023).

However, the flexibility of FSPM comes at the cost of yielding excessive patterns, many of which may be noise (Zhou et al., 2010). This issue increases researchers' burden in interpreting the patterns to obtain actionable insight into the learning process. Interpretation of noisy patterns may waste time and cause incorrect understanding of the learning process, which may lead to ineffective learning design. For example, prior research used FSPM to find learning behaviors patterns that were frequent for high-performing learners and recommend those patterns to low-performing learners (Tarus et al., 2017). Some of these patterns may be frequent due to randomness, and recommending them to low-performing learners may be ineffective. In addition, many patterns found by FSPM may be redundant with each other.

This study addressed the limitations of FSPM by proposing a permutation-based method to remove noisy and redundant patterns. The fundamental assumption is: if a pattern found by a FSPM algorithm in a real event sequence dataset is noise, the pattern would also be found by the algorithm in a randomly generated dataset where the frequencies of individual events are the same as the real dataset. Sequential patterns found in the randomly generated dataset are regarded as noisy patterns. We conducted simulations and empirical data analyses to evaluate the proposed method.

1.1 Notations

Let $S = \{i_1, i_2, \dots, i_n\}$ be a learner's *event sequence*, composing *N* temporally ordered itemset. The subscript number represents the itemset position. An itemset may contain one or more elements from $E = \{e_1, e_2, \dots, e_p\}$, a set of p distinct event types. In educational datasets, an itemset usually is one event (e.g., Wong et al., 2019; Zhou et al., 2010). Thus, this paper uses the terms "itemset" and "event" interchangeably. FSPM uses the gap to denote the position difference between two events. For instance, the gap between i_1 and i_3 is two. Gap is analogous to lag in LSA (Bakeman & Gottman, 1997).

1.2 Frequent sequential pattern mining

FSPM aims to find frequent sequential patterns in a set of sequences (Agrawal & Srikant, 1995). A sequential pattern is a pattern consisting of ordered events, e.g., *copy → paste* and *reading a page → taking a note*. A pattern is frequent if its support value is no less than a pre-specified threshold, where the support value is the proportion of sequences in a dataset that contain at least one occurrence of the sequential pattern (Agrawal & Srikant, 1995). An occurrence of a sequential pattern is a sequence that meet three requirements: (1) the sequence contains all pattern events, (2) the event order in the sequence is the same as the pattern, and (3) the gap between pattern events in the sequence meets users' specifications (Lo et al., 2008). Users may set the minimum and maximum gaps to constrain the property of frequent patterns. If the maximum gap is larger than one, two adjacent pattern events can be at nonconsecutive positions in sequences.

There are hundreds of FSPM algorithms. It is noteworthy that these algorithms produce the same set of patterns, given the same parameter setting (Fournier-Viger et al., 2017). The main differences lie in the execution speed and constraints that can be used.

1.2.1 Recurrent sequential patterns and non-overlapping rules. Traditional FSPM algorithms and the support value solely consider whether a sequence contains at least one occurrence of a sequential pattern, ignoring multiple occurrences. This may lead to a great deal of information loss, especially when sequences are long. Thus, researchers have developed algorithms for mining recurrent sequential patterns, which frequently repeat across and within sequences (Lo et al., 2008). In educational sequence data, the occurrences may be more meaningful than support because it may be rare that students either engage in a learning behavior or not.

While counting the occurrences of a sequential pattern in a sequence, the non-overlapping rule is used by many algorithms (Wu et al., 2020). This rule entails that two occurrences of a pattern can use the same event of a sequence, but this event cannot occupy the same position in both occurrences. Occurrences are considered non-overlapping if this rule is satisfied, and they are treated as a single occurrence if not. For instance, consider a sequence $\{i_1 = A, i_2 = A, i_3 = B, i_4 = A, i_5 = B, i_6 = A\}$ and a maximum gap of two. This sequence seems to have three occurrences of $A \rightarrow B \rightarrow A$: $\{i_1, i_3, i_4\}$, $\{i_2, i_3, i_4\}$ and $\{i_4, i_5, i_6\}$. The first two occurrences overlap because i_3 and i_4 are at the second and third positions in both occurrences. Thus, they are counted as one occurrence. Although all occurrences involve i_4, this event occupies the first position in the third occurrence but the last position in the others. Thus, the third occurrence does not overlap with the others.

1.2.2 Redundant sequential patterns. In sequential pattern mining, there are two special types of patterns: generator patterns and closed patterns. A pattern is a generator if its support is smaller than all of its sub-patterns, while a pattern is closed if its support is larger than all of its super-patterns (Fournier-Viger et al., 2017). For instance, assume $A \rightarrow B$ and $B \rightarrow A$ have only one super-pattern, $A \rightarrow B \rightarrow A$. If the support of $A \rightarrow B$, $B \rightarrow A$, and $A \rightarrow B \rightarrow A$ are .7, .6, .6, respectively, $A \rightarrow B$ is closed, while $B \rightarrow A$ is not closed, and $A \rightarrow B \rightarrow A$ is not a generator. In contrast, if the support of $B \rightarrow A$ increases to .7, it also becomes closed, and $A \rightarrow B \rightarrow A$ becomes a generator. Some algorithms regard the equivalent support values as an indicator of redundancy and only retain either generator patterns or closed patterns. However, a pattern may have the same support but different occurrences with its super-patterns or sub-patterns. Determining the redundancy solely based on support may not be reasonable.

1.3 Permutation tests

Permutation tests determine the *p*-value by comparing test statistics with distributions generated from the observed data (Good, 2000), which is useful when theoretical distribution assumptions do not hold. Assume that we are interested in the difference in knowledge test scores between experimental and control groups. The null hypothesis is no difference in test scores, and the alternative hypothesis is that there is a difference. The most common parametric test for this problem is the independent *t*-test, which computes the test score difference, converts the difference to the *t* statistic, and compares the *t* statistic with a theoretical *t* distribution to accept or reject the null hypothesis. By contrast, the permutation test for this problem includes five steps (Good, 2000):

(1) Analyze the problem to identify the null and alternative hypotheses.

(2) Choose a test statistic. In the example above, the test statistic is the score difference.

(3) Compute the test statistic in the raw data.

(4) Permutate the data and recompute the test statistic for the permutated data. In the example mentioned earlier, the permutation is achieved by randomly reassigning group labels to each student. That is, a student who is actually in the experimental group may be labeled as the control group, and vice versa. The reassignment alters which students a group includes, but the number of students in each group remains the same as in the raw data. The score difference is recalculated after a complete reassignment (where every student is randomly labeled). This step repeats until we obtain test score differences for all possible reassignments, i.e., a permutated distribution of score differences.

(5) Compare the raw score difference with the permutated distribution to accept or reject the hypothesis.

In step 4, the number of possible permutations increases exponentially as the sample size increases. Assume the two groups have the same size, n. The number of possible permutations is $\binom{2n}{n}$. If the sample size is large, it is computationally expensive to execute all permutations to obtain a permutated distribution (an exact permutation test). Alternatively, we may generate a permutated distribution from a subset of all permutations (a sampled permutation test; Bakeman et al., 1996). There are a few ways to obtain the subset, and Monte Carlo sampling is one of the most common ways (Good, 2000). This approach randomly draws a subset of possible permutations. What is a sufficient subset size depends on the context.

A permutation test is unbiased under the exchangeability condition (Hayes, 1996). This condition entails that, under the null hypothesis, observations or variable values on different subjects are exchangeable (Good, 2000). In the example, the exchangeability condition entails that the group labels are exchangeable between any pair of students, which holds when two groups have the same variances of test scores (Romano, 1990).

Previous studies have used the permutation test to examine the associations between two event sequences (Bakeman et al., 1996; Bodner et al., 2021). Researchers also have applied the permutation test to FSPM, as discussed below.

1.4 Mining statistically sound sequential patterns with permutation tests

Traditional pattern mining algorithms focus on searching for patterns in a computationally efficient way. However, many patterns may be noise or false discoveries. This issue restricts the application of pattern mining in many fields (Hämäläinen & Webb, 2019), including education. Statistically sound pattern mining addresses this issue by utilizing statistical tests to control the risk of false discoveries. Statistically sound sequential patterns are those whose frequencies statistically deviate from expected frequencies in a null model (Low-Kam et al., 2013).

Permutation tests have been applied to find statistically sound sequential patterns. Tonon and Vandin (2019) proposed the PROMISE algorithm, which contains two permutation methods: itemset swapping and random permutation. Itemset swapping switches the positions of a random pair of events in the data and repeats the swap many times to generate a permutated dataset. The two events swapped can be from different sequences. Thus, this method assumes a global distribution of event types: the distribution of event types is the same across sequences. In contrast, random permutation restricts the swap between events from the same sequence and does not assume a global distribution, and the event distribution of a sequence is preserved in its permutated counterparts. Tonon and Vandin (2019) compared the two methods and concluded that they complemented each other. Based on itemset swapping, Jenkins et al. (2022) proposed a generic method that is faster than PROMISE and allows users to specify the null model for computing the significance.

However, Pinxteren and Calders (2021) found that any method assuming a global distribution may produce considerably biased results if the assumption is violated. They also indicated that the random permutation method may entail a large amount of Monte Carlo sampling for accurate results and is computationally inefficient. Thus, they proposed PS^2, a polynomial-time algorithm that computes the exact statistical significance of a sequential pattern without Monte Carlo sampling. Both PROMISE and PS^2 conduct permutation tests on frequent patterns identified by traditional FSPM algorithms. In comparison with PROMISE, PS^2 speeds up the computation of permutation tests. However, PS^2 does not apply to the sequential patterns where an event type can occur more than once. This highly limits the application of PS^2 in educational data because a learning event rarely occurs once. For instance, in a MOOC course, a student may watch a lecture video or reattempt a problem multiple times.

2 The current study

Prior studies have used permutation tests to remove noisy sequential patterns, but these studies have determined whether a pattern is noisy merely based on its support value and neglected its multiple occurrences in a single sequence (e.g., Pinxteren & Calders, 2021; Tonon & Vandin 2019). Similarly, previous studies have determined redundancies among frequent patterns solely based on support (Fournier-Viger et al., 2017). The number of occurrences may be more meaningful than the support value because it is rare that students either engage in a behavior or entirely do not. Moreover, a pattern may have a statistically significant support value but an insignificant occurrence value, and vice versa. This study proposes two permutation tests to address these limitations, one for discovering sound patterns with statistically significant occurrences and another for identifying redundancy among these sound patterns in terms of differences in occurrences.

3 Discovering patterns with statistically significant occurrences

3.1 Methodology

Patterns with statistically significant occurrences are those that occur at a rate more than chance. The chance value is computed based on the frequencies of individual events and a null model, where the current event is independent of previous events in a sequence (i.e., a zero-order Markov chain). It is important to note that the expected value may not be zero. Thus, the determination of whether a sequential pattern is noise cannot rely on the differences between its occurrences and zero. We adopt a permutation test methodology following Good's (2000) five steps to address this issue.

3.1.1 Analyze the problem. The problem is determining whether a sequential pattern is noisy or statistically sound. There may be numerous possible sequential patterns in a sequence dataset because any combination of events is a possibility. Most of these possible patterns do not occur in the data, and we do not need to test whether they are noise. Thus, following prior studies using the permutation test in FSPM (Pinxteren & Calders, 2021; Tonon & Vandin, 2019), this study chooses frequent sequential patterns as candidates for permutation tests. The frequent sequential patterns are those whose support value is greater than a prespecified minimum support and can be identified by traditional FSPM algorithms, such as constrained Sequential PAttern Discovery using Equivalence classes (cSPADE; Zaki, 2000) and Prefix-Projected Sequential Pattern Mining (PrefixSpan; Pei et al., 2004).

Let μ denote the observed occurrences of a sequential pattern and μ_0 denote its expected occurrences. The value of μ_0 is calculated under the condition that the current event is independent of prior events, i.e., the event order is random. The null hypothesis is $\mu \leq \mu_0$, and the alternative hypothesis is $\mu > \mu_0$. The null hypothesis means that the observed occurrences in the raw sequences are no more than what we expect to observe in random sequences. Here the test is a one-sided test. If sequential patterns that occur less than chance are also of interest, the test will be two-sided. To date, little research has investigated such sequential patterns in education data. Thus, this study focused on frequent sequential patterns and used a one-sided test.

3.1.2 Choose and compute the test statistic of a sequential pattern in the raw data. The problem is whether a sequential pattern occurs more than chance. Thus, its occurrences will be the test statistic. Both the sum of its occurrences over sequences and the average occurrences per sequence work. However, the interpretation of the average is more straightforward because the sum depends on the number of sequences. Hence, this study uses the average occurrences per sequence as the test statistic.

The number of actions during a learning task varies across learners. Thus, the sequence length varies in empirical data. A long sequence is more likely to contain more occurrences of a particular sequential pattern than a short sequence. That is, the occurrences of a sequential pattern may follow distributions with different centers in sequences with different lengths. The variations in sequence lengths may influence the accuracy of the permutation test. To investigate this impact, this study compared the accuracies between conditions of two test statistics. One was the average occurrences over sequences (Equation (1)). The other was weighted average occurrences, where the inverse of sequence lengths was used as the weights (Equation (2)).

$$\bar{\mu} = \frac{1}{N}\sum_{i=1}^{N} U_i, \tag{1}$$

$$\bar{\mu} = \sum_{i=1}^{N} \frac{\frac{1}{l_i}}{\sum_{j=1}^{N} \frac{1}{l_j}} U_i, \tag{2}$$

N is the number of sequences, and U_i is the observed occurrences of a sequential pattern in the i^{th} sequence. l_i denotes the length of the i^{th} sequence.

Note that using the average occurrences as the test statistic is the main difference between the present study and prior work, which have used the support as the test statistic (e.g., Pinxteren & Calders, 2021; Tonon & Vandin 2019). In subsequent analyses, we compared the performance of the permutation test using average occurrences as the test statistic with that using the support to demonstrate the relative advantages of the proposed method.

3.1.4 Permutate the data and recompute the test statistic. The permutation shuffles events within a sequence. Specifically, given a sequence with *n* events, we randomly sample one event *n* times without replacement from the sequence to generate its permutated counterpart. The position of a sampled event in the permutated sequence is the sampling order. The raw sequence and the permutated only differs in the event order.

It is worth noting that such permutation strategy assumes that the distribution of event types may vary across sequences instead of being homogeneous across sequences. The former assumption is more robust than the latter in mining statistically sound sequential patterns (Pinxteren & Calders, 2021). Also, the former is more realistic in educational data because learners have substantial differences in their engagement with learning activities (e.g., Kinnebrew et al., 2017; Mudrick et al., 2019). Indeed, individual differences in learning process are an important topic in education.

Recall that an unbiased permutation test entails the exchangeability condition, which refers to that the permutation is reasonable under the null hypothesis. In this study, the null hypothesis is that the observed occurrences in raw sequences are no more than the expected occurrences in sequences where event order is random. Under the null hypothesis, it is reasonable to shuffle events. There may be cases in which the learner must execute a particular action before another. For instance, in an intelligent tutoring system, the learner may need to attempt a problem before the hint is available. In this context, swapping the attempt action and the hint-asking action is not reasonable because the exchangeability condition is not met (Hayes, 1996). In this condition, the permutation will underestimate the expected value of the pattern *attempt* → *hint-asking* and inflate the type I error rate, and thus, the permutation test is not suitable for patterns that contain *attempt* → *hint-asking*. Alternatively, we may code adjacent *attempt* and *hint-asking* into a new action before conducting FSPM. Swapping this action and the others is reasonable, and thus, the permutation test can be applied to patterns that contain the new action.

A complete permutation shuffles the event order of all sequences. After each complete permutation, the test statistic will be recomputed using Equation (*1*). Monte

Carlo sampling is used to draw a sample of test statistics $\boldsymbol{\mu_0} = (\mu_0^1, \mu_0^2, \ldots, \mu_0^m)$, where m denotes the number of permutations. Section 3.4 investigates how many permutations are necessary to guarantee accurate results.

3.1.5 Determining whether a sequential pattern is noise. The proportion of permutated test statistics larger than the observed test statistic is computed as the *p*-value:

$$p = \frac{\sum_{j=1}^{m} \mathbf{1}\left[\mu_0^j > \bar{\mu}\right]}{m}, \tag{3}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function, which returns one if the argument is true and zero otherwise. The symbol *m* is the number of permutations. Since each candidate pattern entails a permutation test, the *p*-value needs to be adjusted to control the type I error rate. We may control the familywise error rate or the false discovery rate (FDR). Researchers have recommended controlling the FDR in exploratory research where many patterns need to be tested, and the cost of false discoveries is low (Hämäläinen & Webb, 2019). FSPM is an exploratory method, and there are usually many candidate patterns. Thus, this study controls the FDR via the Benjamini-Yekutieli correction (BY; Benjamini & Yekutieli, 2001). This correction method is used because sequential patterns are likely related to each other. After the BY correction, if a sequential pattern is statistically significant, this pattern will be regarded as a statistically sound pattern. Otherwise, the pattern is noise.

3.2 Simulations for false discovery rates

We applied the proposed permutation test to simulated data to evaluate its FDR and statistical power (in the next section). Specifically, the simulation focused on how the minimum support, maximum gap, sequence length, pattern length, and pattern occurrences influenced the false discovery rate and statistical power. To investigate the advantages of the proposed two test statistics (the average occurrences over sequences and the weighted average occurrences) relative to prior studies, we also analyzed the performance of the test when the support was the test statistic, given that prior studies have focused on it (e.g., Jenkins et al., 2022; Pinxteren & Calders, 2021). cSPADE was used to generate the frequent sequential patterns. It adopts a depth-first strategy for searching frequent patterns, and is fast and scalable while allowing a variety of constraints on the frequent patterns, such as the maximum gap between events that consists of a pattern and the minimum and maximum number of events in a pattern. We used the non-overlapping rule while counting the occurrences of a pattern. The rule restricts that two pattern occurrences can use the same event, but the event cannot be at the same position in different occurrences (Wu et al., 2020).

3.2.1 Data generation. The simulated data were generated by permuting students' action logs in Betty's Brain, a digital environment for learning scientific phenomena (Kinnebrew et al., 2017). The number of event types was 17. As the event order in the simulated data was random, the permutation test should not identify any sequential pattern as statistically sound. Four factors were manipulated: the average sequence length, the number of permutations, the minimum support, and the maximum gap.

The data contained two sequence sets: 98 students' action sequences while learning human body thermoregulation and another 93 students' action sequences while learning about climate change. For the simulation, the number of sequences (i.e., students) had three levels: 20, 50, and 100. Under each level, we randomly selected the corresponding number of sequences from the 191 action sequences.

The average sequence length was 280 in the data, and it varied from tens to hundreds in most educational studies (Chen et al., 2017; Verstege et al., 2023). Thus, the simulation set the average sequence length to four levels: 20, 50, 100, and 300. For instance, when the average length of simulated sequence was 50, we randomly sampled 50/280 events from a raw sequence to form its simulated counterpart. In this way, the simulated data matched the realistic data because the length was heterogeneous across simulated sequences, and the length variation was the same as the real data. When the average sequence length was 300, we randomly sampled 20/280 events from a raw sequence and added the sampled events to the raw sequence to expand it so that the average length of simulated sequences approximated to 300.

The minimum support in prior educational studies varied from .0005 to .97 (Jiang et al., 2015; Mudrick et al., 2019). When the number of participants was 20, the minimum support of .0005 did not make sense because it meant that a sequential pattern was frequent if less than one participant used it. Thus, the minimum support was set to eight levels: .2 to .9, with a step size of .1. Most previous educational studies have set the maximum gap to 1 or 2 (e.g., Kinnebrew et al., 2017; Verstege et al., 2023). The simulation took three levels of the maximum gap (1, 2, 3) to expand the comprehensiveness. Overall, there were four, three, eight, and three levels of the sequence length, the number of sequences, the minimum support, and the maximum gap. The simulation was replicated 1,000 times in each condition.

3.2.2 Analysis. The FDR was computed via Equation (4), where FP_r was the number of statistically sound patterns in the r^{th} replication. As the sequences were randomly generated, any discovered pattern was a false pattern (FP). The term N_r was the number of candidate patterns that were tested in the *r*th replication. Ideally, the FDR should be no more than the significance level used, which was .05. In some conditions, the total number of candidate patterns over the 1,000 repetitions was smaller than 500. This meant that the permutation tests were run less than 500 times in these conditions. Because of the small trials, we regarded the FDRs in the conditions as unreliable and discarded them.

$$FDR = \frac{\sum_{r=1}^{1000} FP_r}{\sum_{r=1}^{1000} N_r}, \tag{4}$$

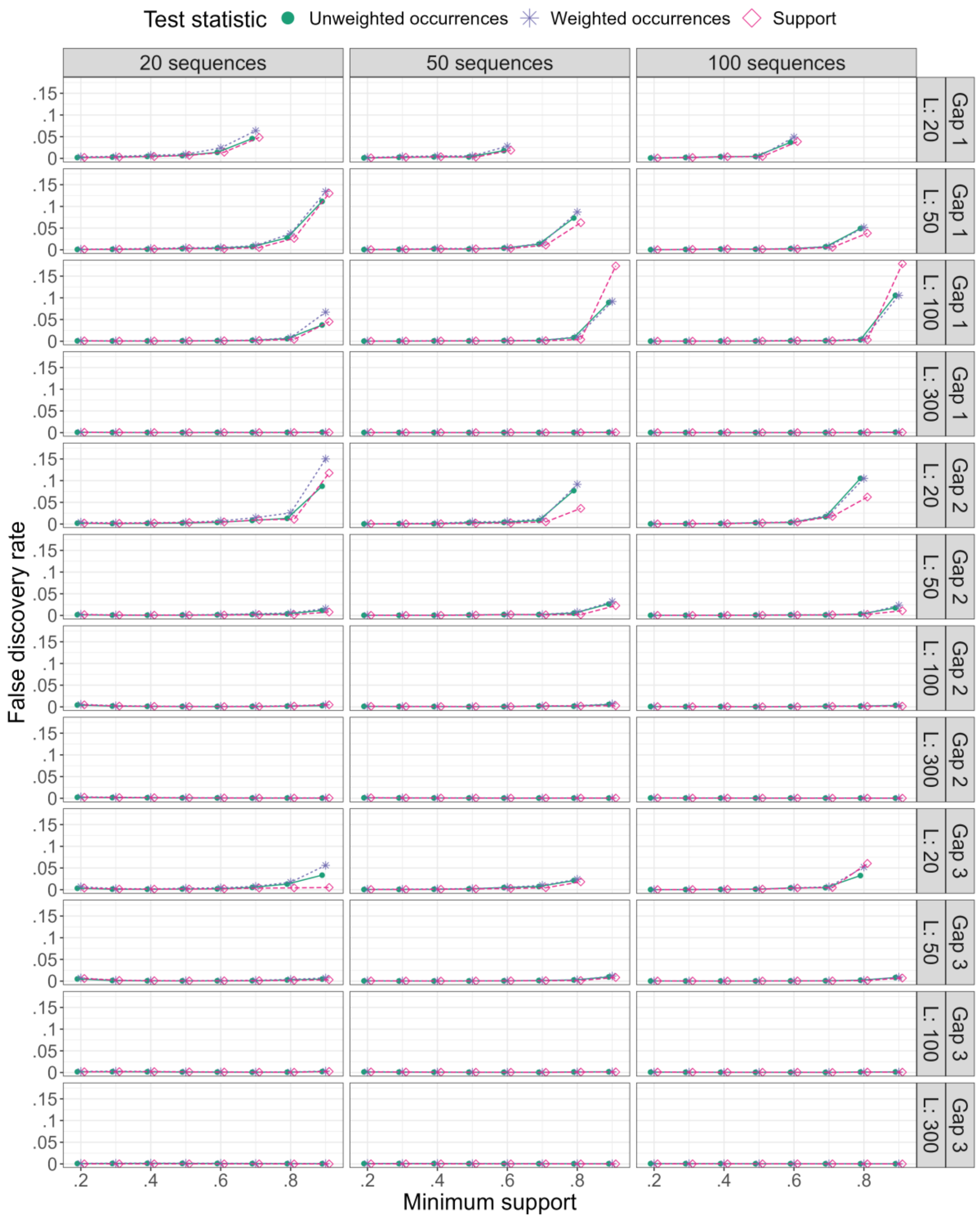
Ten thousand permutations were conducted. We varied the number of used permutations from 200 to 10,000 to investigate the convergence speed of candidate patterns' *p*-values. Results of the convergence speed were reported in section 3.4.

3.2.3 Result. Figure 1 displays the FDR of the permutation test. We first discuss the impact of the sequence length, the number of sequences, the minimum support, and the maximum gap and then compare the performance of different test statistics. The most noticeable is that FDR went up as the support value increased, especially for short sequences. The reason is that in a dataset of short sequences, the support value of a sequential pattern might be strongly related to its average occurrences. For short sequences, most types of events only occurred once or twice; thus, the possible occurrences of the patterns made up of these events were restricted to one or two. For instance, assume that events A and B occur once in a randomly generated sequence of 20 events. When the maximum gap is one, the unique arrangements of events in the sequence would be $\binom{20}{2} * 2!$, representing choosing 2 positions from the 20 positions to hold A and B as well as arranging them in the 2 positions. The arrangements with at least one $A \rightarrow B$ would be $\binom{19}{1}$, representing choosing 1 position from the 19 to hold $A \rightarrow B$. The probability that $A \rightarrow B$ occurs in the sequence would be .05. Assume that there are 20 such sequences, the probability that $A \rightarrow B$ occurs in none of these sequences is $.05^0 *.95^{20} * \binom{20}{0}$, that in one sequence is $0.05^1 * 0.95^{19} * \binom{20}{1}$, that in two sequences is $.05^2 *.95^{18} * \binom{20}{2}$, and so on. If $A \rightarrow B$ has a support value no less than .3 in the 20 sequences, it occurs in at least 6 sequences. A rough estimation of the probability is $1 - \sum_{n=0}^{5} .05^n *.95^{20-n} * \binom{20}{n} = .0003$. Such a low probability means that $A \rightarrow B$ is unlikely to occur in at least six sequences and have high support. Once it does in the raw sequences because of coincidence, the permutation test likely identifies it as statistically sound because the high support is unlikely to occur in the permutations. That is, the permutation test is likely to fail in detecting noisy patterns with high support in short sequences.

The sequence length mitigated the impact of support because the sequence length increased the probability that a pattern occurred in a sequence. Using the beforementioned $A \rightarrow B$ as an example, if the sequence length and counts of A and B events double (40 events, 2 A, and 2 B), the unique arrangements of events in the sequence would be $\binom{40}{2} * 2! * \binom{38}{2} * 2!$, which represents choosing 2 positions from the

Figure 1

False Discovery Rate of Permutation Test



Note. L: the sequence length. Gap: the maximum gap.

40 positions to hold the 2 A events and arranging them in the 2 positions, as well as choosing 2 positions from the remaining 38 positions to hold the 2 B events and arranging them in the 2 positions. The arrangements with at least one $A \rightarrow B$ would be $\binom{2}{1}\binom{2}{1}\binom{39}{1}\binom{38}{2} * 2!$, which represents choosing 1 of the 2 A events and 1 of the 2 B events to form an occurrence of $A \rightarrow B$, choosing 1 position from 39 positions to hold $A \rightarrow B$, choosing 2 positions from the remaining 38 positions to hold the remaining 1 A and B events, and arranging them in the 2 positions. The probability that $A \rightarrow B$ occurs in the sequence would be .1, twice the probability when the sequence length is 20. That is, in longer sequences, high-support patterns are more likely to repeat in permutations. Thus, given the support value, the FDR declined as the sequence length increased.

The maximum gap also moderated the influence of support. The reason is similar to that of sequence length. Using $A \rightarrow B$ in a sequence of 20 events as an example, when the maximum gap is 2, the probability that $A \rightarrow B$ occurs in the sequence is $\frac{\binom{19}{1}+\binom{18}{1}}{\binom{20}{2}2!} \approx$ .10, twice the probability when the maximum gap is 1. Consequently, the probability of $A \rightarrow B$ having high support increases as the maximum gap increases. Thus, we observe that FDR decreased when the maximum gap increased, given the support value.

In contrast, the number of sequences exacerbated the influence of support because the number of sequences decreased the probability that a pattern had a high support. Again, using $A \rightarrow B$ in a sequence of 20 events and the maximum gap of 1 as an example, the probability that $A \rightarrow B$ occurs in a sequence is .05 when the sequence contains only one A and B. Assume that all sequences have 20 events, one A, and one B. The probability that $A \rightarrow B$ has a support no less than .3 is a function of the number of sequences: $1 - \sum_{n=0}^{N*.3-1} .05^{n} * .95^{N-n} * \binom{N}{n}$. The probability decreased from 3.3*10^{-4} to 1.3*10^{-8} as the number of sequences increased from 20 to 50. Thus, we observe that FDR increased when the number of sequences increased and the support was high.

Using the weighted average occurrences increased FDR. The biggest difference between the unweighted and weighted occurrences was .06, which was in the condition of 20 sequences, a maximum gap of 2, an average length of 20, and a minimum support of .9. In 262/274 of the conditions in Figure 1, the differences were smaller than .01.

The performance of support was close to that of unweighted occurrences. In 264/274 of the conditions, the absolute differences were smaller than .01. In five conditions, the FDR of unweighted occurrences were higher than that of support with a difference greater than .01. The biggest difference was .08, in the condition of 50 sequences, a maximum gap of 1, an average length of 100, and a minimum support of .9. In the other five conditions, the FDR of support were higher than that of unweighted occurrences with a difference greater than .01. The biggest difference was .04, in the condition of 100 sequences, a maximum gap of 2, an average length of 20, and a minimum support of .8. Note that the differences in FDR between different test statistics were substantial only in the conditions with a support no less than .8.

3.3 Simulations for statistical power

3.3.1 Data generation. This simulation was based on the permutated Betty's Brain data, but sequential patterns were planted into the permutated sequences (Pinxteren & Calders, 2021). The planted pattern length had three levels, 2, 4, and 6, which matched the range of pattern lengths in prior studies (Mudrick et al., 2019; Verstege et al., 2023).

The average planted occurrences of the sequential pattern in a single sequence had three levels, 0.25, 0.5, and 1.5. When the average occurrences were 0.25 or 0.5, the planted occurrences followed a binomial distribution where the success meant one occurrence. The success probability was .25 when the average occurrences was 0.25, and .5 when the average occurrences was 0.5. When the average occurrences were 1.5, the planted occurrences followed a multinomial distribution (0, 1, 2, 3), where each value had the same probability. In empirical research, the average occurrence may be more than 20 (Kinnebrew et al., 2017). The maximum of the average occurrence was set to 1.5 because a pilot study indicated that 1.5 was sufficient to guarantee a high statistical power, i.e., the sequential pattern would always be identified as statistically sound when the average occurrences were 1.5. More than 1.5 average occurrences do not provide new information. However, note that 1.5 does not guarantee a high power when the sequence characteristics change. In particular, the value of the planted occurrences that guarantees a high power depends on the occurrences of the pattern at chance, which is related to the base rate of pattern events and sequence lengths. If the pattern's events have a high base rate, and sequence length is long, the pattern will have a high occurrence at chance. If 1.5 is much smaller than the occurrence at chance, planting 1.5 occurrences will not result in a high power. The result section will return to this point.

Planting sequential patterns into a sequence increases the sequence length and the frequencies of events that constitute the pattern. To ensure constant sequence lengths and frequencies of pattern events, the number of pattern events equal to that of planted pattern events were removed from the sequence before the planting. For example, if planting two occurrences of $A \rightarrow B$ into a sequence, two A and two B would be randomly selected and removed from the sequence before planting.

The base rates of event types constituting the planted pattern may influence the probability that the planted pattern will be detected as sound. For example, if these event types are frequent, they are likely to appear together to form instances of the planted pattern, even in a randomly generated sequence. Consequently, the occurrences of the planted pattern in a sequence may not be greater than that in the permutated sequence, and the planted pattern may not be identified as statistically sound. Thus, the base rates of event types constituting the planted pattern were manipulated and set to high and low levels. For a high base rate condition, event types with the highest base rates were selected to form the planted pattern. For example, if the planted pattern contains three events, the most frequent three types of events were selected. By contrast, in the low base rate condition, event types with lower base rates were selected to form the planted pattern. Note that event types with the lowest base rates were not always used because a sequence might not contain sufficient events to form occurrences of the planted pattern. The base rates of pattern events in the high base rate condition ranged from .068 to .170, while that in the low base rate condition ranged from .043 to .068.

Five occurrences of a length-6 pattern consume 30 events. If a sequence is short, the sequence likely does not contain at least 30 events. Thus, the average sequence length was fixed to 100. The maximum gap was fixed to 1, while the support was fixed to .5. As we have seen in the simulation result for FDR, when the sequence length was 100, this combination of maximum gap and support assured a low FDR. Also, prior studies on Betty's Brain have mainly used a minimum support of .5 (Emara et al., 2018; Kinnebrew

& Segedy et al., 2017). The number of sequences was fixed to 100. Overall, there were three, three, and two levels of the pattern length, the average pattern occurrence, and base rates of event types, respectively. The simulation replicated 1,000 times under each condition.

3.3.2 Analysis. The statistical power was computed via Equation (5):

$$power = \frac{\sum_{r=1}^{1000} \mathbf{1}|TP_r|}{1000}, \tag{5}$$

where TP_r indicated whether the rth replication identified the planted sequential pattern as statistically sound, i.e., a true positive. Note that within each replication, we tested all patterns with a support value no less than .5 rather than only the planted sequential pattern. The former setting may lead to a lower power of the permutation test in detecting the planted pattern than the latter because the BY correction needs to be used in the former setting. Specifically, in the latter setting, the planted pattern would be labeled as statistically sound if its *p*-value is no more than the raw significance level (.05 in this study). However, in the former setting, the planted pattern needs a much lower *p*-value to be statistically sound because the BY correction adjusts the alpha due to multiple tests. For instance, assume that 50 patterns are tested. The BY correction would be applied to the *p*-values of these patterns. If the planted pattern has the lowest *p*-value among these patterns, the corresponding adjusted alpha is .0002. If its *p*-value is the 25th lowest, the corresponding adjusted alpha is .0056. Although the former setting may lead to lower power, it is more realistic because, in practice, researchers usually apply FSPM in exploratory studies where many patterns are tested. Table 1 presents the average number of patterns tested (i.e., those with a support value no less than .5) in a replication.

Table 1

Average Number of Patterns Tested in a Replication

Base rate	Average planted occurrences	Planted pattern length		
		2	4	6
Low	0.25	42	45	47
	0.5	41	43	47
	1.5	40	39	44
High	0.25	41	43	46
	0.5	41	43	49
	1.5	40	39	43

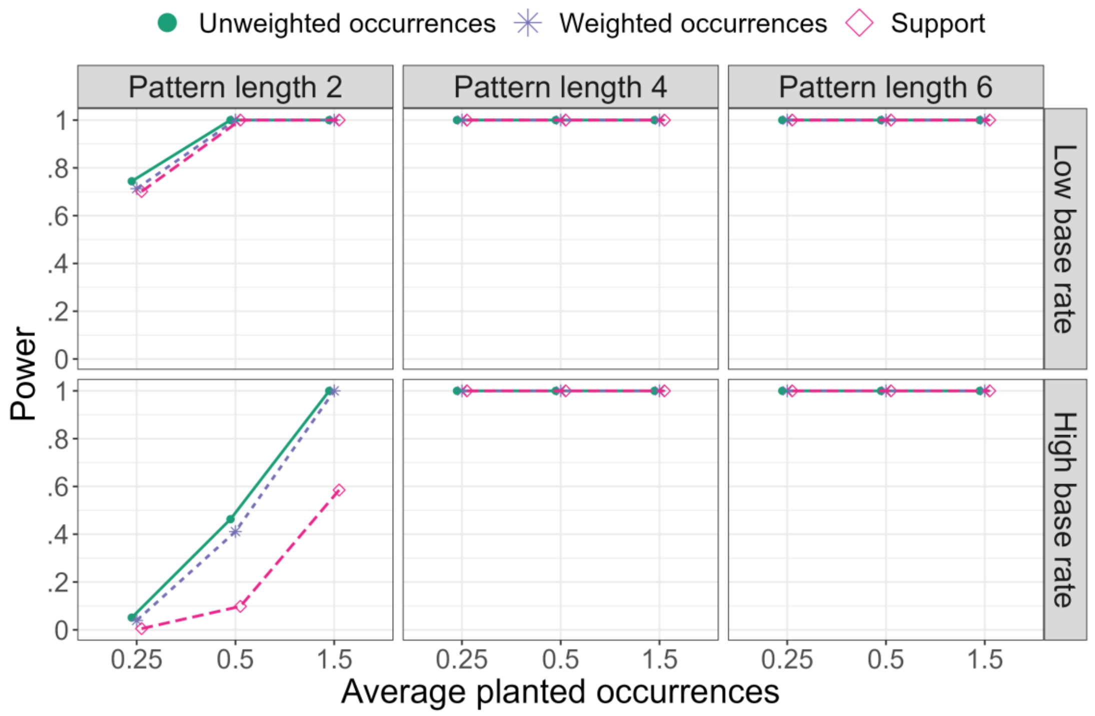
We conducted 10,000 permutations and varied the number of used permutations from 200 to 10,000 to investigate the convergence speed of the planted patterns' *p*-values. Section 3.4 reports the results of the convergence speed.

3*.3.3 Results*. Figure 2 shows that the permutation test with the average occurrences as the test statistic detected the planted patterns with a high statistical power in most conditions (.74 ~ 1.00). The power was low when the length of the planted pattern was 2, the planted average occurrences was 0.25 or 0.5, and the base rate of pattern events was high. In this condition, the short pattern and the high base rate of pattern events together made the pattern likely to occur a few times in a randomly generated sequence. For example, among 10,000 permutations, the planted pattern occurred in a sequence 2.35 times on average. By contrast, when the base rate of pattern events was low, the average occurrences of the planted pattern in a randomly generated sequence were 0.32.

Using the weighted occurrences as the test statistic slightly decreased the statistical power by .01~.05 in conditions where the power was not 1.00. When the average occurrences was the test statistic, regardless of weighted or not, the permutation test had a stastistical power higher than the condition where the support was the test statistic. In particular, the difference was as high as 0.31~0.42 when the length of the planted pattern was 2, the planted average occurrences was 0.5 or 1.5, and the base rate of pattern events was high. The reason was that, in these conditions, the planted pattern already occurred at least one time in most randomly generated sequences. Among 10,000 permutations, the average support was 0.86. Planting the pattern in sequences where it already occurred did not increase its support but did increase its occurrences. That is, the impact of planting the pattern on the average occurrences was stronger than that on the support.

Figure 2

The Power of Permutation Test in Detecting the Planted Patterns



3.4 Convergence speed

Permutation tests are time-consuming. Investigating how the FDR and statistical power change as the number of permutations increases guides selecting the number of permutations to balance accuracy and time cost. This section evaluates the convergence of FDR and statistical power when the test statistic was the unweighted average occurrences, given that the permutation test performed the best with this test statistic. Table 2 presents the selected simulation conditions for investigating the FDR. The first four conditions formed a contrast to evaluate the impact of the sequence length on the convergence of FDR, where the second one was also the simulation condition for statistical power. The fifth to seventh conditions were the same as the fourth but differed in the maximum gap (1 versus 3), the minimum support (.5 versus .2), and the number of sequences (50 versus 100), respectively. The FDR of the above conditions were low. Thus, we choose a condition with a high FDR as the last one.

Table 2

Selected Conditions for Investigating Convergence of the False Discovery Rate

Condition	Sequence length	Maximum gap	Minimum support	# sequences	FDR
1	**300**	1	.5	50	0.0002
2	**100**	1	.5	50	0.0007
3	**50**	1	.5	50	0.0018
4	**20**	1	.5	50	0.0034
5	20	**3**	.5	50	0.0018
6	20	1	**.2**	50	0.0010
7	20	1	.5	**100**	0.0045
8	20	2	.8	50	0.0770

The convergence lines of FDR had different shapes across conditions, but they showed a general trend: a large decline within the first 2,000 permutations (Figure 3). The FDR at the 3,000 permutations was close to the FDR at the 10,000 permutations in these conditions (differences < .003 in the eighth condition and differences < .001 in the others). The convergence of FDR in the fourth, seventh, and eighth conditions fluctuated more than the others, perhaps because the total number of patterns being tested in these three conditions (5040, 4888, 831 patterns, respectively) was smaller than the others (⩾ 12542). The same change in the number of false discoveries resulted in a greater change in FDR in these three conditions than in the others (e.g., 1/831 > 1/12542).

Figure 3

Convergence of the False Discovery Rate of the Permutation Test in Selected Conditions

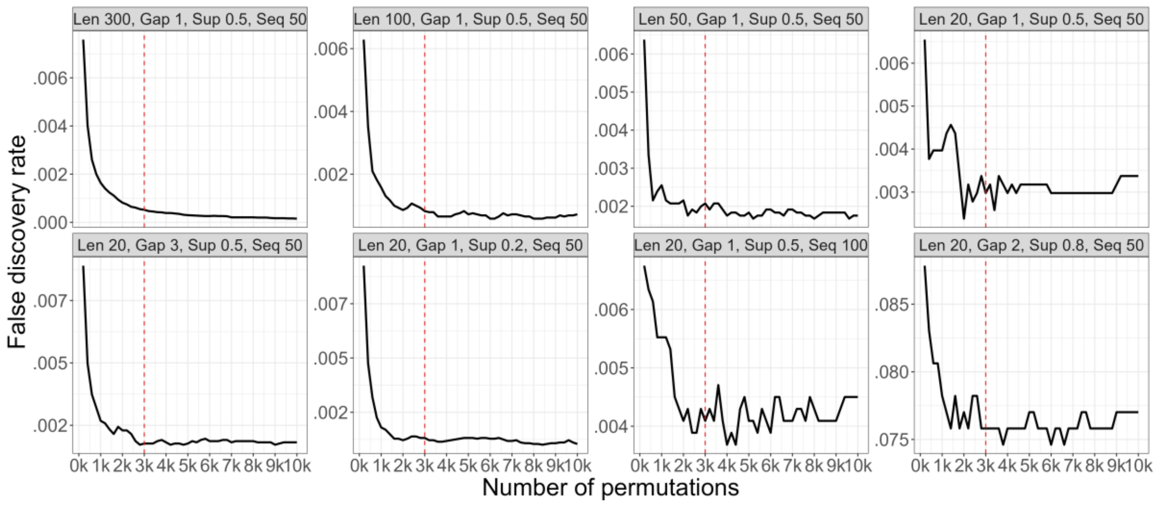
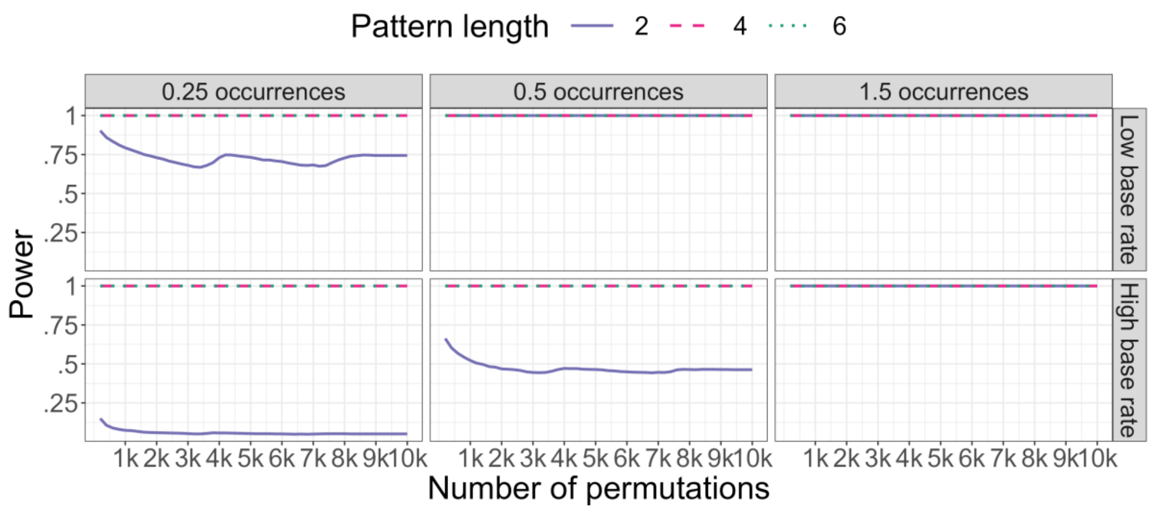


Figure 4 displays the convergence speed of the statistical power. For the conditions where the permutation test always detected the planted pattern, the statistical power was always one regardless of the number of permutations, indicating that the occurrences of the planted patterns in all 10,000 permutations were smaller than the occurrences of the planted patterns in the raw sequences. For the condition where the power was relatively low, when the base rate was high, the statistical power became relatively stable when the number of permutations reached 3,000. When the base rate was low, the statistical power needed 8,000 permutations to be stable. The difference in the power between 3,000 and 8,000 permutations was .063.

Figure 4

Convergence Speed of Statistical Power of the Test for Detecting a Planted Pattern



Surprisingly, the statistical power decreased as the number of permutations increased. This trend was due to the BY correction for multiple tests. On average, there was 43 candidate patterns being tested each time, including the planted pattern. The raw *p*-value of the planted pattern was usually the smallest, and the corresponding adjusted alpha was .00027 based on the BY correction. This means that the *p*-value of the planted pattern would be larger than the adjusted alpha if there was at least one permutation where the occurrences of the planted pattern were larger than its occurrences in the raw sequences, given no more than 3,745 permutations (1/3,745 = .00027). However, as the number of permutations increased from 200 to 3,745, the probability of such a permutation increased. Therefore, the statistical power decreased before stabilizing as the number of permutations increased.

4 Identifying redundant patterns

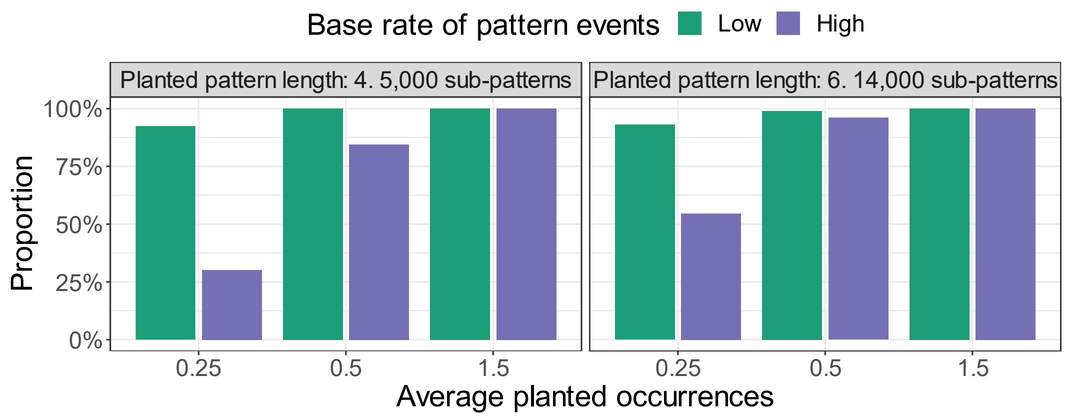
The permutation test in this section aims to identify redundant patterns among statistically sound patterns. Specifically, it focuses on the redundancy between a pattern and its sub-patterns. We used the data generated by the simulation for statistical power to illustrate the necessity of identifying the redundancy and why there is no need to test the redundancy between a pattern and its super-patterns.

After planting a sequential pattern, the occurrences of its super-pattern might increase. For instance, planting an occurrence of $A \rightarrow B$ between C and D in a sequence also results in one occurrence of $C \rightarrow A \rightarrow B$, $A \rightarrow B \rightarrow D$, and $C \rightarrow A \rightarrow B \rightarrow D$. However, the planting increased occurrences of different super-patterns in different sequences because the position for planting was random. Thus, it was expected that the permutation test would not identify these super-patterns as statistically sound. Indeed, across the 18,000 repetitions (18 conditions × 1,000 repetitions per condition), the permutation test only labeled 145 super-patterns of the planted pattern as statistically sound. Thus, a long pattern is unlikely being statistically sound because of its sub-patterns.

Planting a sequential pattern longer than two increased its sub-patterns. For instance, planting an occurrence of $A \rightarrow B \rightarrow C$ implies planting an occurrence of $A \rightarrow B$ and $B \rightarrow C$ (and $A \rightarrow C$, if the maximum gap > 1). Thus, the permutation test may identify sub-patterns as statistically sound. Indeed, only in the conditions of 0.25 average planted occurrence and high base rates of planted events, a small proportion of sub-patterns of the planted pattern were labeled as statistically sound (30% ~ 55%). In the other conditions, over 84% of sub-patterns of the planted pattern were labeled as statistically sound (Figure 5). Labeling the sub-patterns as statistically sound was not a false discovery because the sub-patterns were indeed planted, although not intentionally. Nevertheless, the sub-patterns were redundant because they did not contain more information than the planted pattern. This issue is possible in reality: learners may execute long behavior patterns; consequently, the sub-patterns of the long patterns are also executed and may be identified as statistically sound. Thus, it is necessary to identify such redundant patterns to simplify the set of statistically sound patterns.

Figure 5

The Proportions of Statistically Sound Sub-Patterns of the Planted Patterns.



4.1 *Methodology*

4.1.1 Analyze the problem. The problem is, for a statistically sound pattern, to determine whether it is sound because of itself and its super-pattern or merely the super-pattern. That is, when excluding its occurrences attributed to the super-pattern, do the remaining occurrences still differ from the chance value? Let μ denote the occurrences of a sequential pattern and μ_{sup} denote the occurrence of its super-pattern in the raw data. $d = \mu - \mu_{sup}$ is the occurrence difference, i.e., the occurrences of a pattern without the effect of its super-pattern. Let d_0 be the expected occurrence difference at chance. The null hypothesis is $d \leq d_0$, while the alternative hypothesis is $d > d_0$. We need the distribution of d_0 to test the null hypothesis. In a randomly permutated sequence dataset, the occurrence difference between a pattern and its super pattern is due to chance. Hence, we can utilize permutation to construct the distribution of d_0 under the null hypothesis.

4.1.2 Choose and compute the test statistic in the raw data. We use the difference in the average occurrences between a pattern and its super-pattern as the test statistic. In the raw data, it is computed as $\bar{d} = \bar{\mu} - \overline{\mu_{sup}}$, where $\bar{\mu}$ and $\overline{\mu_{sup}}$ are the average occurrences per sequence of the pattern and its super-pattern computed via Equation (1).

4.1.3 Permutate the data and recompute the test statistic. The permutated occurrences for identifying statistically sound patterns can be used to compute the test statistic in permutated data. Consequently, the test statistics in different permutations form the permutated distribution of the test statistic, $\mathbf{d_0} = (d_0^1, d_0^2, \dots, d_0^m)$, where m is the number of permutations.

4.1.4 Determine whether a sequential pattern is redundant. The proportion of permutated value larger than the observed value is computed as the p-value:

$$p = \frac{\sum_{j=1}^{m} \mathbf{1}\left[d_0^j > \bar{d}\right]}{m}, \tag{6}$$

A short pattern may correspond to multiple super-patterns, and a long pattern may have multiple sub-patterns. However, it is unnecessary to test each pair of patterns and super-patterns. If the occurrence difference between a pattern and one super-pattern is not significant, it is redundant, and there is no need to test its redundancy with the other super-patterns. This implies that testing the redundancy between a pattern and all its sub-patterns simultaneously is more efficient than testing the redundancy between a pattern and all its super-patterns simultaneously. This is because the latter may include unnecessary tests. Although we may test the redundancy one super-pattern at a time, we cannot apply correction to multiple tests, potentially inflating the type I error rate.

The non-redundancy between a pattern and its sub-patterns is monotone. Assume a pattern *P* of length *l* had a sub-pattern P_{l-1} of length *l-1*, which in turns has a sub-pattern P_{l-2} of length *l-2*, and so on, until it reaches a sub-pattern P_2 of length two. If P_{l-1} is not redundant with *P*, then P_{l-2}, …, P_3 and P_2 will not be redundant with *P*, because $\overline{\mu_{P_2}} \geq \cdots \geq \overline{\mu_{P_{l-2}}} \geq \overline{\mu_{P_{l-1}}} > \overline{\mu_P}$. Because of the monotonic property, we only need to test the redundancy between a pattern and its sub-patterns with one fewer event. This study applied the BY correction to the multiple tests between a pattern and its one-fewer-event sub-patterns. After the correction, if the occurrence difference between a pattern and a one-fewer-event sub-pattern is not significant, the sub-pattern is considered redundant. There is no need to test the sub-pattern's redundancy with the other super-patterns.

4.2 Simulations

4.2.1 Data generation. We evaluated the proposed redundancy test in two settings, where a pattern was statistically sound because of its super-pattern and where a pattern was statistically sound because of both itself and its super-pattern. The simulated data in section 3.3 were used for the condition where a pattern was statistically sound because of its super-pattern. This condition was denoted as the *raw* setting.

To construct the setting that a pattern was statistically sound because of both itself and its super-pattern, we randomly added zero or one occurrence for each sub-pattern of the planted pattern into every sequence of the *raw* setting. That is, we added approximately 0.25 to the average occurrences per sequence for each sub-pattern. Consequently, the occurrence difference between a sub-pattern and the planted pattern increased by approximately 0.25. We named the new setting *additional* setting. Within each of the 1,000 repetitions in both settings, the statistical significance of the occurrence differences between the planted pattern and its sub-patterns were computed using the permutation test described above. The proportion of the repetitions where the permutation test yielded a significant result was calculated. In the *raw* setting, the proportion was an indicator of the false discovery rate, while in the *additional* setting, the proportion was an indicator of the statistical power. Ideally, this proportion should be close to zero in the *raw* setting and close to one in the *additional* setting.

4.2.2 Results. Figure 6 shows that in the *raw* setting, the proportion was zero in all conditions, suggesting that the permutation test could identify patterns that were statistically sound because of its super-pattern. For the *additional* setting, the proportion was one in most conditions, suggesting that the permutation test was unlikely to label patterns as redundant that were statistically sound because of itself and its super-pattern. Exceptions happened in the condition where the sub-pattern length was 2, and the base rates of events in the planted pattern were high. This is expected because the short sub-patterns and the high base rate of pattern events together made the sub-pattern likely to occur a few times in a randomly permutated sequence. That is, the occurrence difference between the sub-pattern and the planted pattern was relatively large even at chance. Adding 0.25 to the occurrence difference in the raw data did not distinguish it much from the occurrence difference due to chance.

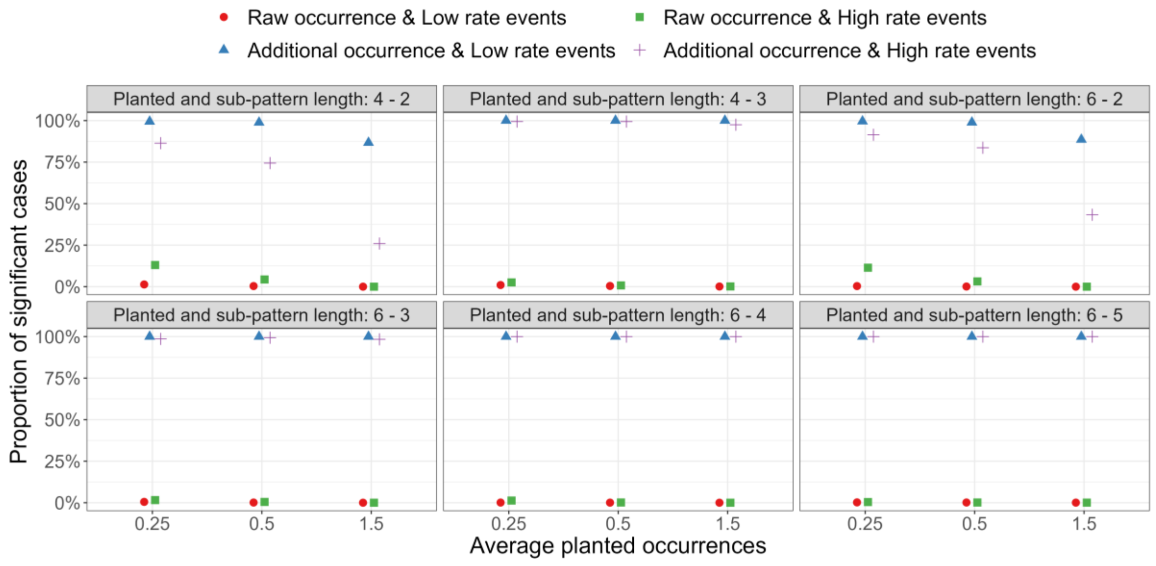
5 Empirical data analyses

We conducted empirical data analyses to investigate three research questions (RQs).

RQ1: Are the support values of statistically sound and noisy patterns different? RQ1 evaluates the advantage of the proposed method in comparison to the support values of sequential patterns. The permutation test is time-consuming. If the difference in support is large between statistically sound and noisy patterns, researchers may use a high support threshold to ensure that most of the obtained sequential patterns are statistically sound. But if the difference is trivial, a high threshold may discard both noisy and statistically sound patterns. In the latter condition, the permutation test becomes necessary.

Figure 6

The Proportion of Repetitions with Significant Differences



Note. Low/high rate events: the base rates of events in the planted pattern were low/high.

RQ2: Is a statistical sound (or noisy) pattern in one dataset likely sound (or noisy) in another dataset? RQ2 evaluates the generalizability of patterns' statistical soundness, akin to the overfitting issue of machine learning models. If patterns' statistical soundness identified by the permutation test cannot be generalized to new sequences, the permutation test loses practical significance.

RQ3: Do redundant sub-patterns identified by the permutation test have the same support as their super-pattern? RQ3 compares the permutation-based redundancy detection method with related work, which has determined redundant patterns merely based on the support value. If the relationship of support between a sub-pattern and its super-pattern accurately indicates the redundancy identified by the permutation method, the permutation method would lose its advantages and be unnecessary.

5.1 Data processing

We used four educational datasets: Betty's Brain, LabBuddy, iSnap, and Jo Wilder. The Betty's Brain dataset contained the data for simulations and a new sample's data. The LabBuddy dataset was graduates' action logs in LabBuddy, a virtual experiment environment (Verstege et al., 2023). The iSnap dataset was undergraduates' logs in iSnap, an intelligent block-based programming application (Price et al., 2017). The Jo Wilder dataset was players' logs in Jo Wilder and the Capitol Case, a game for learning history (PBS Wisconsin Education, 2019). Table 3 displays the characteristics of these datasets.

Table 3

Characteristics of Empirical Datasets

Learning environment	Learner type	# Sequences	Average sequence length (standard deviation)	# Event types
Betty's Brain	Sixth graders	191	280 (147)	17
LabBuddy	Graduate	81	577 (202)	18
iSnap	Undergraduates	54	2,896 (1,541)	65
Jo Wilder	Diverse	23,568	625 (102)	21

To assess the generalizability of patterns' statistical soundness, we split each dataset into training data and test data. The training data of the Betty's Brain dataset were 98 students' action logs while learning about human body thermoregulation. The test data were another 93 students' action logs while learning about climate change. For the other datasets, the training and test data were obtained via randomly splitting a dataset into two halves with equal numbers of learners.

5.2 Analysis

The cSPADE algorithm was applied to the training dataset to find frequent sequential patterns (Zaki, 2000). Prior studies have applied FSPM to investigate the frequent behavior patterns in Betty's Brain and LabBuddy (Kinnebrew & Segedy et al., 2017; Verstege et al., 2023). We followed these studies to set the minimum support (.5 and .3) and the maximum gap (two and one) in the analyses on Betty's Brain and LabBuddy datasets. For the other datasets, we set the maximum gap as one and varied the minimum support, which were .5 and .7 for iSnap and Jo Wilder datasets, respectively. In this way, the parameter settings for Betty's Brain and iSnap data formed a contrast in the maximum gap; that for LabBuddy, iSnap, and Jo Wilder formed a contrast in support.

The permutation tests described in sections 3 and 4 were applied to identify which candidate patterns were statistically sound or noisy in training and test data, respectively. The test statistic was the unweighted average occurrences. The raw significance level was .05, and BY correction was applied to control the false discovery rate. The independent *t* test was used to examine the differences in interestingness metrics between statistically sound and noisy patterns. Cohen's *kappa* was used to evaluate the generalizability of patterns' statistical soundness.

5.3 Results and discussion

Figure 7 displays the distributions of support values of statistically sound and noisy patterns. In all datasets, the support values of sound patterns were not different from noisy patterns ($p \geqslant .34$). These results indicate that using high support does not guarantee that the obtained patterns are sound.

Figure 7

The Differences in Support between Sound and Noisy Patterns

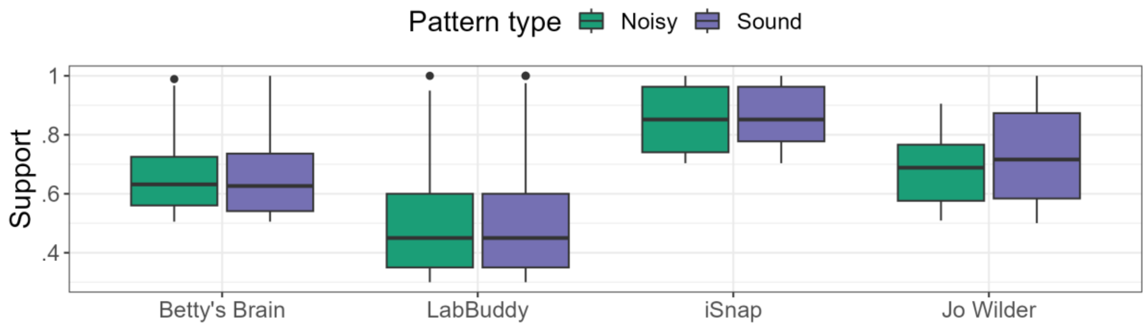


Table 4 displays the number of statistically sound and noisy patterns in the training and test data. In all datasets, over 93% of frequent patterns were either sound in both training and test data or noisy in both data, and *kappa* was .784 ~ .883. Thus, the statistical soundness of patterns generalized well to new datasets.

Table 4

Statistically Sound and Noisy Patterns in the Training and Test Datasets

Training dataset (counts & proportion)		Test dataset		*kappa*
		Sound	Noisy	
Betty's Brain	Sound (386)	368 (76%)	18 (4%)	.784
	Noisy (98)	16 (3%)	82 (17%)	
LabBuddy	Sound (326)	316 (79%)	10 (3%)	.851
	Noisy (73)	8 (2%)	65 (16%)	

iSnap	Sound (817)	884 (90%)	14 (1%)	.883
	Noisy (166)	5 (1%)	80 (8%)	
Jo Wilder	Sound (331)	331 (97%)	0 (0%)	1.000
	Noisy (10)	0 (0%)	10 (3%)	

Among the sound patterns in training data, 35.7%, 9.2%, 67.8%, and 81.0% were identified as redundant in Betty’s Brain, Labbuddy, iSnap, and Jo Wilder datasets, respectively. Among the redundant patterns, only 3.6%, 16.7%, 21.5%, and 0.0% had the identical support values with their super patterns in the four datasets, respectively. The mean differences (and standard deviations) in support values between redundant patterns and their super-patterns were .10 (.10), .12 (.10), .03 (.06), and .05 (.12). The results thus suggest the necessity of using permutation tests to detect redundancy.

Table 5 presents example patterns in Betty’s Brain. The first pattern indicates that students read an e-book page after receiving a prompt recommending reading specific text, while the second pattern indicates that students quickly read a page after receiving a prompt. A student exhibited the first pattern 4.01 times on average, but the pattern was not sound because its permutated occurrences were as high as 3.78. By contrast, the second pattern was sound because its average occurrences were 1.37 higher than its permutated occurrences. Nevertheless, the second pattern was redundant with the third one, which was redundant with the fourth. The fourth pattern meant that, in Betty’s Brain, students took a quiz, received a prompt for reading, read one page less than ten seconds, and read another page no less than ten seconds. This finding suggests that, although students followed the prompt for reading, they sometimes had difficulty in finding the page immediately that contained the text recommended by the prompt, as indicated the fact that *Quick read* occurred before *Read*.

Table 5

Example patterns in Betty's Brain

Patterns	Occurrences	Permutated	Sound	Redundant
1. Prompt → Read	4.01	3.78	N	-
2. Prompt → Quick read	4.45	3.08	Y	Y
3. Prompt → Quick read → Read	2.82	0.91	Y	Y
4. Quiz → Prompt → Quick read → Read	2.22	0.15	Y	N

Note. Prompt: students received a prompt for reading a specific e-book page. Quick read: reading a page for less than 10 seconds. Read: reading a page for no less than 10 seconds. Quiz: students took a quiz.

6 Overall discussion

This study proposed permutation tests for detecting statistically sound and nonredundant sequential patterns among those found by traditional FSPM algorithms. Simulations indicated that the permutation test for detecting sound patterns had a low false discovery rate and high power in most conditions. The permutation test for detecting redundant patterns also showed a high accuracy. Using the average occurrences as the test statistic resulted in higher statistical power versus using the support as the test statistic, which was observed by prior work (e.g., Jenkins et al., 2022; Pinxteren & Calders, 2021). We applied the tests to four empirical datasets of learning event sequences and found that the results of the permutation test from one dataset were generalizable to another dataset. Overall, the analyses showed that the proposed permutation test effectively identifies sound and nonredundant sequential patterns.

Statistical soundness is a statistical property of patterns and may not guarantee researchers' interest (Geng & Hamilton, 2006). A statistically sound pattern of learning events is not necessarily meaningful. Nevertheless, they are at least more likely to capture learning behaviors or strategies than noisy patterns, and interpretation on them may be more reliable. Additionally, removing noisy and redundant patterns reduces the number of patterns to be interpreted and eases the burden of interpretation.

6.1 Practical implications for sequential pattern mining and the permutation test

Studies have applied FSPM for understanding behavioral differences within individual and collaborative learning environments, evaluating the effectiveness of pedagogical interventions, and enriching theoretical problem-solving models. Regardless of which purpose, reliable results are necessary. The necessity of reliable results highlights the importance of identifying and removing noisy patterns. The finding of this study suggests that tuning the parameters of traditional FSPM algorithms can reduce the number of frequent behavioral patterns but is not an effective method for removing noisy behavioral patterns. With a high support threshold and a small maximum gap, there may still be tens or hundreds of noisy patterns that meet the criteria. Moreover, using high support to remove noisy patterns may risk missing meaningful behavioral patterns because whether a behavioral pattern occurs at a rate more than chance is weakly related to these metrics. By contrast, applying the permutation test proposed by this study can identify most of the noisy patterns, meanwhile with a high power of detecting behavioral patterns that are likely to be meaningful. The permutation test performed better when the test statistic was unweighted average occurrences of a pattern than when it was support and weighted average occurrences. Therefore, we recommend applying the permutation test with unweighted average occurrences to frequent patterns found by traditional FSPM algorithms before interpretation and further analyses.

In order to remove redundant patterns to simplify the set of frequent patterns, some FSPM algorithms discard patterns if they have the same support value with their sub-patterns or super-patterns (Fournier-Viger et al., 2017). However, the same support should not be the only criterion for discarding patterns. Discarding patterns should be based on whether their occurrence information is redundant with other patterns. In the empirical data analyses, most of the sub-patterns of the planted pattern had support values higher than the planted pattern itself, but the sub-patterns and the planted pattern were redundant. Thus, equivalent support is not an accurate indicator of redundant patterns. Instead, using the permutation test proposed by this study is a better approach for identifying redundant patterns.

One inevitable question while applying a permutation test is how many permutations to use. With more permutations, the precision increases, but the time cost also increases, particularly when the number of candidate patterns is large. For instance, the number of candidate patterns in the Betty's Brain data in the empirical data analyses was 484, and it took nearly 15 hours to complete 10,000 permutations. For the permutation test proposed by the current study, we recommend 3,000 permutations. As shown in Figure 3 and Figure 4, the FDR and power at 3,000 permutations were close to that at 10,000 permutations. Some may question whether 3,000 permutations is sufficient for long sequences because the number of possible permutations of a long sequence is much larger than a short sequence. However, this is likely not the case. As the first row of Figure 3 displays, the convergence is similar across the levels of the average sequence length.

6.2 Limitations and future research

The permutation test showed a high false discovery rate when sequences were short, and support was high (see Figure 1). The reason is that a sequential pattern with high support is likely to have many occurrences in a dataset of short sequences. In these conditions, researchers should be cautious with patterns labeled as statistically sound by the proposed permutation test. This issue is worthy of future research because short sequences are not rare in education. For future directions, we suggest avoiding parametric tests, which assume the distributions of events in a sequence and compute the probability that a pattern occurs at certain times based on the distribution. Such tests also likely have high false discovery rates (see the first paragraph about results in section 3.2).

The permutation test for identifying redundant patterns assumes that all sound patterns are ground truth. But the permutation test for identifying sound patterns has type I and II errors, which may influence the permutation test for identifying redundant patterns. For example, a super-pattern of one sound pattern may be labeled as statistically sound incorrectly, and the subsequent test may fail in finding a significant difference between the sub-pattern and the super-pattern, and thus label the sound sub-pattern as redundant. Future research may explore approaches that identify the redundancy in the sound patterns while accounting for the potential statistical errors of these patterns.

The permutation test proposed by this study is time-consuming. Even 3,000 permutations still take hours to complete. The most time-consuming part is counting pattern occurrences, whether in the raw or the permutated data. The time of this part depends on the number of patterns to be counted and the maximum gap. When the maximum gap is larger than one, counting pattern occurrences takes extra time to check whether any occurrence of a pattern overlaps with the other occurrences (i.e., the same position of these occurrences uses the same event of a sequence). By contrast, two pattern occurrences never use the same event at the same position when the maximum gap is one. One solution for reducing the time is parallel programming (Gan et al., 2019) because permutations are independent and can be processed parallelly. Another one is using more efficient algorithms to count pattern occurrences. Recently developed FSPM algorithms count pattern occurrences efficiently (Wu et al., 2020). Further work may integrate them with the permutation test proposed by the current study.

The permutation test assumes the exchangeability condition, where any event can occur at any time. Such a condition likely holds in sequences where the event is, for example, coded discourse or think-aloud data because learners may say any words at any time theoretically. However, for action logs in digital learning environments, it is likely that some actions are conditional on others (e.g., in an environment where a learner can only take a quiz for some topic after completing the reading for that topic). The issue may be mitigated by coding the two actions into a single event when they are adjacent. Nevertheless, such a coding solution requires researchers' deep understanding of the learning environment. Future research may develop new methods to address the issue. For instance, while analyzing the synchrony between two time series, Bodner et al. (2021) found that swapping segments of time series instead of individual time points resulted in more accurate results because swapping segments accounts for the auto-dependency within a time series. Auto-dependency is analogous to situations where one action is conditional on another. Thus, swapping event dyads rather than single event may circumvent the exchangeability assumption, but it entails further investigations on whether swapping event dyads alters the statistical power of the permutation test.

References

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In P. S. Yu & A. L. P. Chen (Eds.), *Proceedings of the 11th international conference on data engineering* (pp. 3-14). IEEE Computer Society.

Bakeman, R., Robinson, B. F., & Quera, V. (1996). Testing sequential association: Estimating exact p values using sampled permutations. *Psychological Methods*, *1*(1), 4-15.

Bakeman, R., & Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis* (2nd ed.). Cambridge university press.

Benjamini, Y., & Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, *29*(4), 1165-1188.

Bodner, N., Tuerlinckx, F., Bosmans, G., & Ceulemans, E. (2021). Accounting for auto-dependency in binary dyadic time series data: A comparison of model- and permutation-based approaches for testing pairwise associations. *British Journal of Mathematical and Statistical Psychology*, *74*, 86-109.

Chen, B., Resendes, M., Chai, C. S., & Hong, H. (2017). Two tales of time: Uncovering the significance of sequential patterns among contribution types in knowledge-building discourse. *Interactive Learning Environments*, *25*(2), 162-175.

Emara, M., Rajendran, R., Biswas, G., Okasha, M., & Elbanna, A. A. (2018). Do students' learning behaviors differ when they collaborate in open-ended learning environments? *Proceedings of the ACM on Human-Computer Interaction, 2*(CSCW), 49.

Fournier-Viger, P., Lin, J. C., Kiran, R. U., Koh, Y. S., & Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, *1*(1), 54-77.

Gan, W., Lin, J. C., Fournier-Viger, P., Chao, H., & Yu, P. S. (2019). A survey of parallel sequential pattern mining. *ACM Transactions on Knowledge Discovery From Data*, *13*(3), 25.

Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys*, *38*(3), 9.

Good, P. (2000). *Permutation tests: A practical guide to resampling methods for testing hypotheses* (2nd ed.). Springer.

Hämäläinen, W., & Webb, G. I. (2019). A tutorial on statistically sound pattern discovery. *Data Mining and Knowledge Discovery*, *33*(2), 325-377.

Hayes, A. F. (1996). Permutation test is not distribution-free: Testing $H_0: \rho = 0$. *Psychological Methods*, *1*(2), 184-198.

Hopcan, S., Polat, E., & Albayrak, E. (2022). Collaborative behavior patterns of students in programming instruction. *Journal of Educational Computing Research*, 1420786676.

Jenkins, S., Walzer-Goldfeld, S., & Riondato, M. (2022). SPEck: Mining statistically-significant sequential patterns efficiently with exact sampling. *Data Mining and Knowledge Discovery*, *36*(4), 1575-1599.

Jian, P., Jiawei, H., B., M., Jianyong, W., H., P., Qiming, C., U., D., & Mei-Chun, H. (2004). Mining sequential patterns by pattern-growth: The PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, *16*(11), 1424-1440.

Jiang, Y., Paquette, L., Baker, R. S., & Clarke-Midura, J. (2015). Comparing novice and experienced students within virtual performance assessments. In O. C. Santos, J. G. Boticario, C. Romero, M. Pechenizkiy, A. Merceron, P. Mitros, J. M. Luna, C. Mihaescu, P. Moreno, A. Hershkovitz, S. Ventura, & M. Desmarais (Eds.), *Proceedings of the eighth international conference on educational data mining* (pp. 136-143). International Educational Data Mining Society.

Kinnebrew, J. S., Segedy, J. R., & Biswas, G. (2017). Integrating model-driven and data-driven techniques for analyzing learning behaviors in open-ended learning environments. *IEEE Transactions on Learning Technologies*, *10*(2), 140-153.

Knight, S., Wise, A. F., & Chen, B. (2017). Time for change: Why learning analytics needs temporal analysis. *Journal of Learning Analytics*, *4*(3), 7-17.

Lo, D., Khoo, S., & Liu, C. (2008). Efficient mining of recurrent rules from a sequence database. In J. R. Haritsa, R. Kotagiri, & V. Pudi (Eds.), *Database systems for advanced applications. DASFAA 2008. Lecture notes in computer science, vol 4947* (pp. 67-83). Springer.

Low-Kam, C., Raïssi, C., Kaytoue, M., & Pei, J. (2013). Mining statistically significant sequential patterns. In H. Xiong, G. Karypis, B. Thuraisingham, D. Cook, & X. Wu (Eds.), *IEEE 13th international conference on data mining* (pp. 488-497). IEEE.

Molenaar, I. (2014). Advances in temporal analysis in learning and instruction. *Frontline Learning Research*, *2*(4), 15-24.

Molenaar, I., & Wise, A. F. (2022). Temporal aspects of learning analytics - grounding analyses in concepts of time. In C. Lang, G. Siemens, A. F. Wise, D. Gašević, & A. Merceron (Eds.), *The handbook of learning analytics* (2nd ed., pp. 66-76). SoLAR.

Mudrick, N. V., Azevedo, R., & Taub, M. (2019). Integrating metacognitive judgments and eye movements using sequential pattern mining to understand processes underlying multimedia learning. *Computers in Human Behavior*, *96*, 223-234.

PBS Wisconsin Education. (2019). Jo Wilder and the Capitol Case. Retried from https://pbswisconsineducation.org/jowilder/about/

Pinxteren, S., & Calders, T. (2021). Efficient permutation testing for significant sequential patterns. In C. Demeniconi & I. Davidson (Eds.), *Proceedings of the 2021 SIAM international conference on data mining (SDM)Proceedings* (pp. 19-27). Society for Industrial and Applied Mathematics.

Price, T. W., Dong, Y., & Lipovac, D. (2017). iSnap: Towards Intelligent Tutoring in Novice Programming Environments. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 483-488). Association for Computing Machinery.

Reimann, P. (2009). Time is precious: Variable- and event-centred approaches to process analysis in CSCL research. *International Journal of Computer-Supported Collaborative Learning*, *4*(3), 239-257.

Robertson, A. M., & Willett, P. (1998). Applications of n‐grams in textual information systems. *Journal of Documentation, 54*(1), 48-67.

Romano, J. P. (1990). On the behavior of randomization tests without a group invariance assumption. *Journal of the American Statistical Association*, *85*(411), 686-692.

Soderstrom, N. C., & Bjork, R. A. (2015). Learning versus performance. *Perspectives On Psychological Science*, *10*(2), 176-199.

Tarus, J. K., Niu, Z., & Yousif, A. (2017). A hybrid knowledge-based recommender system for e-learning based on ontology and sequential pattern mining. *Future Generation Computer Systems*, *72*, 37-48.

Tonon, A., & Vandin, F. (2019). Permutation strategies for mining significant sequential patterns. In J. Wang, K. Shim, & X. Wu (Eds.), *Proceedings of 2019 IEEE international conference on data mining* (pp. 1330-1335). IEEE.

Ulitzsch, E., He, Q., & Pohl, S. (2022). Using sequence mining techniques for understanding incorrect behavioral patterns on interactive tasks. *Journal of Educational and Behavioral Statistics*, *47*(1), 3-35.

Verstege, S., Zhang, Y., Wierenga, P., Paquette, L., & Diederen, J. (2023). Using Sequential Pattern Mining to Understand How Students Use Guidance While Doing Scientific Calculations. *Technology, Knowledge and Learning*

Wong, J., Khalil, M., Baars, M., de Koning, B. B., & Paas, F. (2019). Exploring sequences of learner activities in relation to self-regulated learning in a massive open online course. *Computers & Education*, *140*, 43-56.

Wu, Y., Zhu, C., Li, Y., Guo, L., & Wu, X. (2020). NetNCSP: Nonoverlapping closed sequential pattern mining. *Knowledge-Based Systems*, *196*, 105812.

Zaki, M. J. (2000). Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management* (pp. 422-429). Association for Computing Machinery.

Zhou, M., Xu, Y., Nesbit, J. C., & Winne, P. H. (2010). Sequential pattern analysis of learning logs: Methodology and applications. In C. Romero, S. Ventura, M. Pechenizkiy, & R. S. J. D. Baker (Eds.), *Handbook of Educational Data Mining* (pp. 107-121). CRC Press.

Zhang, Y., & Paquette, L. (2023). Sequential pattern mining in educational data: The application context, potential, strengths, and limitations. In A. Peña-Ayala (Ed.), *Educational data science: Essentials, approaches, and tendencies* (pp. 219-254). Springer.