Efficiently Sampling Interval Patterns from Numerical Databases

Djawad Bekkoucha[a], Lamine Diop[b], Abdelkader Ouali[a], Bruno Crémilleux[a], Patrice Boizumault[a]

[a] *Université de Caen Normandie, ENSICAEN, CNRS, Normandie Univ, GREYC UMR6072, 6 Boulevard Marechal Juin, Caen, F-14000, France*
[b] *EPITA Research Laboratory (LRE), Le Kremlin-Bicetre, Paris, FR-94276, France*

Abstract

Pattern sampling has emerged as a promising approach for information discovery in large databases, allowing analysts to focus on a manageable subset of patterns. In this approach, patterns are randomly drawn based on an interestingness measure, such as frequency or hyper-volume. This paper presents the first sampling approach designed to handle interval patterns in numerical databases. This approach, named FIPS, samples interval patterns proportionally to their frequency. It uses a multi-step sampling procedure and addresses a key challenge in numerical data: accurately determining the number of interval patterns that cover each object. We extend this work with HFIPS, which samples interval patterns proportionally to both their frequency and hyper-volume. These methods efficiently tackle the well-known long-tail phenomenon in pattern sampling. We formally prove that FIPS and HFIPS sample interval patterns in proportion to their frequency and the product of hyper-volume and frequency, respectively. Through experiments on several databases, we demonstrate the quality of the obtained patterns and their robustness against the long-tail phenomenon.

Keywords: Numerical data, Data mining, Pattern sampling, Interval patterns.

1. Introduction

Data scientists have a central role for knowledge discovery from data. In practice, analysts want to interact (visualise, select, explore) not only with the data, but also with the patterns or models supported by the data. To

carry out such processes, it is essential to produce high quality results within a very short time so that the analyst does not have to wait. Otherwise, there is a risk that they will disconnect from the system. Pattern sampling is one solution to this challenge [1].

Pattern sampling aims at randomly selecting a pattern with a probability proportional to an interestingness measure [2, 3, 4] , such as frequency. For example, a pattern $\mathcal{V}_1$ that is twice as frequent as a pattern $\mathcal{V}_2$ will be twice as likely to be selected. A naive approach to pattern sampling is to generate the entire set of patterns and then perform a weighted draw based on the interestingness measure. However, in practice, this fails due to the large size of the search space, even for simple pattern languages such as itemsets, i.e. data described by Boolean values. An answer to this problem was provided by Boley et al. [4], who defined an approach based on two successive draws. Once the problem has been properly decomposed, this approach guarantees an exact draw proportionally to the distribution resulting from the interestingness measure. Itemsets are the patterns considered in [4].

Numerical data are present in a wide range of applications such as transcriptome analysis [5], medicine [5] and energy [6]. Even if handling numerical data to define patterns goes back to the origins of data mining [7], pattern mining in numerical data remains a challenging task. A simple approach to cope with numerical data is to reuse existing methods for discrete data by first converting data into a binary representation [8]. However, it is well-known that the binarization process often leads to a loss of information.

Kaytoue at al. [9] introduced interval patterns, which preserve all the original information. However, this framework faces scalability issues due to the large number of generated patterns, which highlights the importance of pattern sampling in this context. To the best of our knowledge, there is only one sampling method designed for numerical data [10]. This method uses a metric to construct neighbourhood patterns whose relevance is characterized by a density measure. Data are considered continuous and the method requires the setting of a parameter that defines the size (diameter) of a pattern.

Our work focuses on sampling in numerical data. We strive to keep the complete original information expressed by the numerical data by using *Interval Patterns* [9]. We present FIPS and HFIPS, two exact and non-enumerative methods for sampling interval patterns. FIPS samples interval patterns proportionally to their frequency, while HFIPS extends this by incorporating hyper-volume, resulting in patterns sampled proportionally to the product

of their frequency and hyper-volume. The key challenge for both methods is to compute, for each object in the database, its contribution to the overall sampling distribution. For example, in the case of the frequency, the main challenge is to determine the exact number of interval patterns covering an object and without enumerating all covering patterns. This allows us to bias the sampling toward patterns with high interestingness scores. We formally prove that FIPS and HFIPS sample patterns proportionally to the desired distributions. Experimentally, we evaluate the quality of the obtained patterns by using several criteria (frequency, hyper-volume times frequency, impact of the long tail phenomenon, diversity, speed, and plausibility). FIPS and HFIPS are the first methods for interval pattern sampling.

The paper is organized as follows. Section 2 introduces the notations, definitions, and the problem statement. Section 3 discusses related work on pattern sampling. Section 4 presents the FIPS method, while Section 5 extends it with HFIPS. Finally, Section 6 reports an experimental evaluation of the proposed approaches.

2. Preliminaries

2.1. Numerical Dataset

A *numerical dataset* $\mathcal{N}$ is defined by a set of objects $\mathcal{G}$ where each object is described by a set of attributes $\mathcal{M}$. Each attribute $m \in \mathcal{M}$ has a range $\mathcal{N}_m$ which is a finite set containing all the values of the data occurring in attribute m. An object $g \in \mathcal{G}$ is defined by a vector of numerical values $< v_{g,m} >_{\forall m \in \mathcal{M}}$. A dataset where the values of all attributes are binary $\mathcal{N}_m = \{0, 1\}, \forall m \in \mathcal{M}$, is a special case of a numerical dataset and referred as a *binary dataset*.

Example 1. *Table 1 shows a running example of a numerical dataset containing 5 objects* $\mathcal{G} = \{g_1, g_2, g_3, g_4, g_5\}$*, each object is described by 3 attributes* $\mathcal{M} = \{m_1, m_2, m_3\}$.

2.2. Interval Patterns

Patterns in numerical datasets can be represented in many ways, we use the notion of Interval Pattern [9] which is defined as a vector of intervals $\mathcal{V} = \langle [a_m, b_m] \rangle_{\forall m \in \mathcal{M}}$, where $a_m, b_m \in \mathcal{N}_m$ and $a_m \leq b_m$. Each dimension of the vector $\mathcal{V}$ corresponds to an attribute following a canonical order on

	m_1	m_2	m_3
g_1	2	8	130
g_2	4	12	102
g_3	3	7	91
g_4	2	9	101
g_5	6	12	110

Table 1: A running example of a numerical dataset $\mathcal{N}$

the set of attributes $\mathcal{M}$. We denote $\mathcal{B}[g] = \langle [v_{g,m}, v_{g,m}] \rangle_{\forall m \in \mathcal{M}}$ as the vector of intervals corresponding to an object identified by g. An object g is an occurrence of the interval pattern $\mathcal{V}$ if each interval in the vector $\mathcal{B}[g]$ is included in the interval of $\mathcal{V}$, i.e. $\mathcal{B}[g] \sqsubseteq \mathcal{V} \iff [v_{g,m}, v_{g,m}] \subseteq [a_m, b_m], \forall m \in \mathcal{M}$. The cover of $\mathcal{V}$ in $\mathcal{N}$ is the set of objects $g \in \mathcal{G}$ occurring in $\mathcal{V}$, i.e. $cover(\mathcal{V}) = \{g \in \mathcal{G} \mid \mathcal{B}[g] \sqsubseteq \mathcal{V}\}$.

Example 2. *In the example dataset of Table 1,* $\mathcal{V} = \langle [3,4], [7,12], [91,130] \rangle$ *is an interval pattern covering the objects* $\{g_2, g_3\}$. $\mathcal{B}[g_2] = \langle [4,4], [12,12], [102,102] \rangle$ *is the vector of intervals identified by the object* g_2 *and an occurrence of* $\mathcal{V}$.

The frequency of $\mathcal{V}$ is the cardinality of its cover, i.e. $freq(\mathcal{V}) = |cover(\mathcal{V})|$. Given a minimum frequency threshold θ, the interval pattern $\mathcal{V}$ is frequent if and only if $freq(\mathcal{V}) \geq \theta$. The smallest description of a subset of objects $G \subseteq \mathcal{G}$ is the smallest interval pattern covering the set of objects G. Formally the smallest description of G is the interval pattern $\mathcal{V}$ such that for each $g \in G$, g is an occurrence of $\mathcal{V}$, i.e. $desc(G) = \langle [a_m, b_m] \rangle_{\forall m \in \mathcal{M}}$ such that $a_m = min(\{v_{g,m} \mid g \in G\})$ and $b_m = max(\{v_{g,m} \mid g \in G\})$. The hyper-volume of an interval pattern $\mathcal{V} = \langle [a_m, b_m] \rangle_{\forall m \in \mathcal{M}}$ corresponds to the product of the lengths of its intervals, i.e. $vol(\mathcal{V}) = \prod_{[a_m, b_m] \in \mathcal{V}} (b_m - a_m)$.

Note that for an interval pattern $\mathcal{V}$, if the interval associated to an attribute $m \in \mathcal{M}$ has its lower bound equal to the smallest possible value of $\mathcal{N}_m$ and its upper bound equal to the highest possible value of $\mathcal{N}_m$, then the interval is unconstrained. Otherwise, it is considered constrained on the subset of values it contains.

Let $\mathcal{L}$ be the language of interval patterns, which corresponds to the set of all possible interval patterns. The size of the search space is the product of the total number of possible intervals for each attribute. Formally this is

given by:

$$\prod_{m \in \mathcal{M}} \sum_{k=1}^{|\mathcal{N}_m|} k = \prod_{m \in \mathcal{M}} \frac{|\mathcal{N}_m|(|\mathcal{N}_m|+1)}{2}$$

Consider the dataset presented in Table 1, which contains 5 objects and 3 attributes. Each of the attributes m_1 and m_2 contains 4 distinct values, respectively $\mathcal{N}_{m_1} = \{2, 3, 4, 6\}$ and $\mathcal{N}_{m_2} = \{7, 8, 9, 12\}$, and attribute m_3 contains 5 distinct values, namely $\mathcal{N}_{m_3} = \{91, 101, 102, 110, 130\}$. The total number of interval patterns in the search space is given by :

$$\underbrace{\frac{4 \times 5}{2}}_{m_1} \times \underbrace{\frac{4 \times 5}{2}}_{m_2} \times \underbrace{\frac{5 \times 6}{2}}_{m_3} = 1500 \text{ interval patterns}$$

This example illustrates the size of the interval pattern search space. An exhaustive enumeration of all these patterns would be infeasible both in terms of time and memory. Moreover, given the large number of patterns, identifying the relevant ones would become a challenging task. This motivates the use of pattern sampling to explore the search space more efficiently.

2.3. Problem Statement

Let Ω be a population and $f : \Omega \longrightarrow [0, 1]$ a measure. The notation $x \sim f(\Omega)$ denotes that the element x is drawn randomly from Ω with a probability distribution $\pi(x) = f(x)/\sum_{x' \in \Omega} f(x')$.

Given a numerical database $\mathcal{N}$, the language of interval patterns $\mathcal{L}$ and $k \in \mathbb{N}$, the problem of sampling interval patterns consists in retrieving k patterns $\mathcal{V}_1, \ldots, \mathcal{V}_k$ from $\mathcal{L}$, where each pattern $\mathcal{V}_i$ is randomly drawn with replacement with a probability proportional to its value of f. Formally, this corresponds to:

$$Sampling_k(\mathcal{L}, \mathcal{N}, f) = \bigcup_{i=1}^{k} \{\mathcal{V}_i \in \mathcal{L} |\ \mathcal{V}_i \sim f(\mathcal{V}_i)\}$$

This work focuses on the frequency measure, i.e. $freq(\mathcal{V}_i)$, and demonstrates how the principle can be extended to the product of hyper-volume and frequency, i.e. $vol(\mathcal{V}_i) \times freq(\mathcal{V}_i)$.

3. Related Work

We have organized the overview of the pattern sampling methods into three families: probabilistic, declarative and multi-step methods.

The first family is based on Markov chain Monte Carlo algorithms, where the target sampling distribution corresponds to the stationary distribution of a random walk. These methods can handle various pattern languages and interestingness measures. For example, Hasan and Zaki [2] introduced the first pattern sampling approach that handles a graph pattern language and is biased towards a frequency measure. Similarly, Boley et al. [11] proposed sampling formal concepts using strictly positive interestingness measures, while Bendimerad et al. [12] focused on sampling tiles proportionally to a subjective interest. Camelin et al. [13] define a compression-based approach leveraging the LCM algorithm [14] to sample frequent and closed itemsets. Finally, Opran et al. [15] introduced SIMAS, an interactive approach for sampling classification rules according to user-defined interest. Despite their accuracy, stochastic approaches often suffer from slow convergence.

The second family is based on declarative paradigms. FLEXICS [3] uses a SAT solver to sample itemsets and considers a wide range of measures. This approach randomly partitions the search space into different cells by recursively generating XOR constraints on variables associated with the items describing the patterns. However, modeling other pattern languages is challenging, as the encoding needs to be adapted for each language to ensure efficient implementation.

The third family is composed of multi-step methods. These methods partition the occurrences of the patterns into well-devised groups, enabling group sampling based on their weights, and then uniformly sampling a pattern within the selected group. The challenge is to find an appropriate multi-step decomposition to obtain the desired distribution. This approach was pioneered by Boley et al. [4], using a two-step draw to sample patterns according to the frequency. Diop et al. [16] extend this approach with a three-step method for sequential pattern sampling, using a frequency measure and a maximum length constraint. Soulet [17] combines the sampling and constrained-based paradigms by designing an itemset pattern sampling approach enforcing a minimum frequency constraint. Diop [18] proposed a two-step procedure to sample high average utility itemsets under length constraints. Finally, Diop and Plantevit [19] extended this work to sampling high-utility patterns from knowledge graphs by converting them into quan-

titative transactional databases.

Richer patterns such as subgroups are also addressed by pattern sampling. In this context, Moens and Boley [20] present a weighted controlled pattern sampling method for exceptional model mining. This approach divides the search space and applies targeted weighting, balancing efficiency and relevance to discover exceptional patterns.

To the best of our knowledge, there is only one method for output space pattern sampling from numerical data [10]. This method uses a multi-step approach. In contrast to FIPS and HFIPS, it considers a continuous data space, samples according to a density measure, and requires the pattern size to be specified. Our work falls into this class of approaches due to its ability to directly obtain the desired sampling distribution.

4. Sampling Interval Patterns Proportionally to their Frequencies

This section introduces FIPS, an approach for sampling interval patterns proportionally to their frequency. The problem is formally defined as follows:

$$Sampling_k(\mathcal{L}, \mathcal{N}, freq) = \bigcup_{i=1}^{k} \{\mathcal{V}_i \in \mathcal{L} \mid \mathcal{V}_i \sim freq(\mathcal{V}_i)\}$$

4.1. FIPS Key Ideas

To draw a pattern proportionally to its frequency, it is necessary to determine the sum of the frequencies of all patterns in the solution space, i.e. $\sum_{\mathcal{V} \in \mathcal{L}} \text{freq}(\mathcal{V})$. Due to the size of the solution space $\mathcal{L}$, it is intractable to compute this sum by enumerating each solution in the space. This limitation can be overcome by using a multi-step sampling procedure [4]. The key idea is to compute weights by grouping patterns according to their occurrences in the set of objects. Each group is then drawn in proportion to its calculated weight, then a pattern is selected uniformly within the chosen group. Determining the number of patterns covering an object in binary data is straightforward as it is simply equal to $2^{|g|}$, where $|g|$ is the number of items present in the object g. In contrast, for numerical data, this task is more complex. Unlike itemsets, which are contained within each transaction, interval patterns are not necessarily contained within the objects. Moreover, computing the exact number of interval patterns covering an object g requires, for each attribute, considering all possible intervals that include the values appearing in g. To address this challenge, we define a function NIP

(Number of Interval Patterns) that computes the exact number of interval patterns covering an object.

Using this function, it is possible to calculate the sum of the frequencies of all patterns (the normalisation constant Z, see Section 4.3) by considering only the patterns which cover an object. Thus, this method computes a probability distribution proportional to frequency without enumerating the entire solution space.

4.2. Counting the Number of Interval Patterns Covering an Object

This section defines the function $NIP : g \to \mathbb{N}$ which counts the exact number of interval patterns covering the object $g \in \mathcal{G}$. For a value $v_{g,m} \in \mathcal{N}_m$ appearing in an object $g \in \mathcal{G}$, we define:

- $I(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \leq v_{g,m}\}$ as the set of distinct values in attribute m that can be used as lower bounds in interval patterns covering the object g.
- $J(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \geq v_{g,m}\}$ as the set of distinct values in attribute m that can be used as upper bounds in interval patterns covering the object g.

The $NIP(g)$ function is defined as follows:

$$NIP(g) = \prod_{m \in \mathcal{M}} |I(v_{g,m})| \cdot |J(v_{g,m})| \tag{1}$$

Equation 1 relies on two terms. The first term $|I(v_{g,m})|$ counts the number of distinct values in $\mathcal{N}_m$ that can serve as lower bounds for intervals associated to attribute m. Similarly, the second term $|J(v_{g,m})|$ counts the number of distinct values in $\mathcal{N}_m$ that can serve as upper bounds for intervals associated to attribute m. The product $|I(v_{g,m})| \cdot |J(v_{g,m})|$ returns, for an object $g \in \mathcal{G}$ and an attribute $m \in \mathcal{M}$, the total number of intervals including the value $v_{g,m}$. The exact number of interval patterns covering the object g is given by the product of the possible intervals across all attributes in the database.

Example 3. *Consider the numerical database $\mathcal{N}$ presented in Table 1, the object $g_3 \in \mathcal{G}$, and the attribute $m_1 \in \mathcal{M}$. The distinct value of attribute m_1 in object g_3 is 3. The intervals associated to m_1 that include the value 3 are the intervals $[2,3]$, $[2,4]$, $[2,6]$, $[3,3]$, $[3,4]$ and $[3,6]$. Thus, the total number of intervals including the value 3 for the attribute m_1 is 6. Referring to the*

terms in Equation 1, we have: $|I(3)| \cdot |J(3)| = |\{2,3\}| \cdot |\{3,4,6\}| = 6$. *The total number of interval patterns covering* g_3 *is obtained by multiplying the number of intervals for each attribute (see Equation 1):* $6 \times 4 \times 5 = 120$.

4.3. FIPS Sampling Algorithm

This section describes the FIPS sampling procedure (see Algorithm 1). The algorithm starts by determining the number of interval patterns covering each object in the database $\mathcal{N}$ (line 3). This is performed by using the NIP function (see Section 4.2). Line 4 implements the first step of the two-step sampling procedure: an object g is sampled with a probability proportional to the number of interval patterns covering g. Thus, the draw is biased to select patterns covering a high number of objects. Then, in the second step (line 5), an interval pattern covering at least the object g is built attribute by attribute through successive interval samplings. Specifically, for each attribute $m \in \mathcal{M}$, two values $a_m \in I(v_{g,m})$ and $b_m \in J(v_{g,m})$ are uniformly sampled within each respective set. These values form the interval associated to attribute m and are concatenated into the pattern under construction (line 10). Once all attributes have been associated to an interval, the resulting interval pattern $\mathcal{V}$ is returned (line 12). A sample of k patterns is obtained by running k times Algorithm 1.

4.4. Sampling Distribution

In this section, we demonstrate that FIPS samples interval patterns in proportion to their frequencies.

Property 1. *For a numerical database* $\mathcal{N}$*, Algorithm 1 samples an interval pattern* $\mathcal{V}$ *proportionally to its frequency.*

Proof. Let $\mathcal{L}$ be the space of all interval patterns, and $Z = \sum_{\mathcal{V} \in \mathcal{L}} |cover(\mathcal{V})|$ be the normalization constant, representing the sum of the frequencies of all interval patterns. Let $g^* \in \mathcal{G}$ be an object randomly drawn from step 1 of Algorithm 1, and $\mathcal{V}^*$ be an interval pattern sampled in step 2 of the same algorithm. Then:

Algorithm 1: Sampling an interval pattern proportionally to frequency (FIPS)

Input : A numerical database $\mathcal{N}$;
Output : An interval pattern $\mathcal{V}$ sampled proportionally to its frequency ;

Pre-processing: Compute $w_F(g) = NIP(g)$ for each object $g \in \mathcal{G}$;

Step 1: $g \sim w_F(g)$; `/* Draw an object g proportionally to its weight` w_F `*/`

Step 2: ; `/* Draw uniformly an interval pattern` $\mathcal{V}$ `covering` g. `*/`
$\mathcal{V} \longleftarrow \langle\rangle$;
foreach *attribute* $m \in \mathcal{M}$ **do**
  $a_m \longleftarrow$ Draw uniformly a value from $I(v_{g,m})$;
  $b_m \longleftarrow$ Draw uniformly a value from $J(v_{g,m})$;
  $\mathcal{V} \longleftarrow \mathcal{V} + +[a_m,\ b_m]$; `/* Concatenate the interval` $[a_m, b_m]$ `to` $\mathcal{V}$ `(++ denotes concatenation) */`

Return $\mathcal{V}$

$$
\begin{aligned}
P[\mathcal{V}^* = \mathcal{V}] &= \sum_{g \in \mathcal{G}} P[\mathcal{V}^* = \mathcal{V} \ \cap \ g^* = g] \\
&= \sum_{g \in \mathcal{G}} P[\mathcal{V}^* = \mathcal{V} \mid g^* = g] \ \cdot \ P[g^* = g] \\
&= \sum_{g \in \ cover(\mathcal{V})} \frac{1}{NIP(g)} \frac{NIP(g)}{Z} \\
&= \sum_{g \in \ cover(\mathcal{V})} \frac{1}{Z} = \frac{|cover(\mathcal{V})|}{Z} = \frac{freq(\mathcal{V})}{Z}
\end{aligned}
$$

where $Z = \sum_{g \in \mathcal{G}} NIP(g)$ (which is equal to $Z = \sum_{\mathcal{V} \in \mathcal{L}} |cover(\mathcal{V})|$) □

4.5. Complexity Analysis of FIPS

In this section we conduct a computational complexity analysis for FIPS that considers both data pre-processing and the sampling procedure.

Property 2. *The overall time complexity of the FIPS approach for sampling a single interval pattern is:*

$$O(|\mathcal{G}| \cdot |\mathcal{M}| + log|\mathcal{G}| + |\mathcal{M}|)$$

Proof. The numerical database $\mathcal{N}$ contains $|\mathcal{G}|$ objects and $|\mathcal{M}|$ attributes. In the worst case, each attribute has $|\mathcal{G}|$ distinct values. The time complexity of the pre-processing step, which uses the NIP function, is $O(|\mathcal{G}| \cdot |\mathcal{M}|)$. This corresponds to computing the candidate sets $\mathcal{U}(v_{g,m})$ and $\mathcal{A}(v_{g,m})$ for each of the $|\mathcal{M}|$ attributes, where each computation takes $O(|\mathcal{G}|)$ time.

The time complexity of sampling an interval pattern using Algorithm 1 combines the cost of drawing an object in step 1 and sampling intervals for each attribute in step 2. Drawing an object proportionally to its $NIP(g)$ value is done by computing the cumulative sum of all NIP values, generating a random number scaled by the total sum, and locating the corresponding object via dichotomic search in $O(\log |\mathcal{G}|)$. Sampling intervals uniformly in step 2 takes $O(|\mathcal{M}|)$. Therefore, the overall asymptotic time complexity, including both the pre-processing and sampling phases, is $O(|\mathcal{G}| \cdot |\mathcal{M}| + \log |\mathcal{G}| + |\mathcal{M}|)$.

□

Sampling k interval patterns requires $O(k(\log |\mathcal{G}| + |\mathcal{M}|))$ time in addition to the one-time pre-processing cost of $O(|\mathcal{G}| \cdot |\mathcal{M}|)$.

In an interactive pattern mining process, the analyst is often interested in patterns that cover a substantial region of the search space. For interval patterns, this corresponds to patterns with a large hyper-volume. However, considering only hyper-volume may lead to patterns covering only few observations. To avoid such undesirable patterns, we propose to consider both hyper-volume and frequency. The next section presents HFIPS, an extension of FIPS, which samples patterns proportionally to the product of their frequency and hyper-volume.

5. HFIPS: Incorporating Hyper-volume to FIPS

This section presents HFIPS, a method that samples patterns proportionally to the product of their frequency and hyper-volume. The problem is formally defined as follows:

$$Sampling_k(\mathcal{L}, \mathcal{N}, freq \times vol) = \bigcup_{i=1}^{k} \{\mathcal{V}_i \in \mathcal{L} \mid \mathcal{V}_i \sim freq(\mathcal{V}_i) \cdot vol(\mathcal{V}_i)\}$$

5.1. HFIPS Key Ideas

HFIPS is built on the same core principles as FIPS (cf. Section 4.1). It computes, for each pattern in the solution space, the product of its hyper-volume and frequency without explicitly enumerating the entire space. This allows HFIPS to obtain the total sum $\sum_{\mathcal{V} \in \mathcal{L}} (freq(\mathcal{V}) \cdot vol(\mathcal{V}))$ while avoiding a full enumeration. To achieve this, we introduce a new function, denoted IPH, which takes as input an object $g \in \mathcal{G}$ and computes the sum of the hyper-volumes of the patterns covering g. Summing the IPH values across all objects leads to the normalization constant Z, as each pattern contributes proportionally to its frequency (see Section 6.1).

5.2. Efficiently Counting Total Hyper-volumes

This section defines the function $IPH : g \to \mathbb{N}$, which computes the sum of the hyper-volumes of interval patterns covering the object $g \in \mathcal{G}$. IPH is based on the idea of summing the lengths of all intervals that contain an object. We start by introducing this notion.

Total Intervals Lengths. To efficiently compute the hyper-volumes of interval patterns covering an object $g \in \mathcal{G}$, we first consider, for each attribute $m \in \mathcal{M}$, the sum of the lengths of all intervals containing the value $v_{g,m} \in \mathcal{N}_m$. To this end, we use the sets $I(v_{g,m})$ and $J(v_{g,m})$ introduced previously:

- $I(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \leq v_{g,m}\}$ as the set of distinct values in attribute m that can be used as lower bounds in interval patterns covering the object g.
- $J(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \geq v_{g,m}\}$ as the set of distinct values in attribute m that can be used as upper bounds in interval patterns covering the object g.

The TIL (Total Interval Lengths) function can be defined as follows:

$$TIL_m(g) = \left(\sum_{x \in J(v_{g,m})} x \cdot |I(v_{g,m})| \right) - \left(\sum_{x \in I(v_{g,m})} x \cdot |J(v_{g,m})| \right) \quad (2)$$

where, the first term ($\sum_{x \in J(v_{g,m})} x \cdot |I(v_{g,m})|$) represents the sum of interval lengths by summing directly all the values of the upper bounds in $J(v_{g,m})$ and multiplying this sum by the number of lower bounds in $I(v_{g,m})$ since all upper bounds will be combined with each lower bound to form the intervals on the attribute including $v_{g,m}$. However, this term alone overestimates the total length of all intervals, since it includes the upper bounds without subtracting the corresponding lower bounds that define the actual interval lengths. On the opposite side, the second term ($\sum_{x \in I(v_{g,m})} x \cdot |J(v_{g,m})|$), corrects this overestimation by subtracting the sum of interval lengths by summing directly all the values of the lower bounds in $I(v_{g,m})$ and multiplying this sum by the number of upper bounds in $J(v_{g,m})$. Thus, ensuring with the first term that the length of each interval is computed exactly once. Consequently, the TIL function leads to the exact sum of the lengths of all intervals containing $v_{g,m}$ over the attribute m.

Example 4. *Consider the numerical dataset $\mathcal{N}$ shown in Table 1. For simplicity, we focus on attributes m_1 and m_2. The Total Intervals Length for the object g_1 is computed for both attributes as follows:*

- *Total interval lengths for m_1: The value of g_1 for m_1 is 2. $TIL_{m_1}(g_1)$ computes the total length of all intervals that contain the value 2. Here, $I(2) = \{2\}$ and $J(2) = \{2, 3, 4, 6\}$. The first part, $|I(2)| \cdot \sum J(2) = 1 \cdot (2 + 3 + 4 + 6)$, sums all possible upper bounds for intervals having 2 as a lower bound, assuming they can all be paired with this single lower bound. However, this overestimates the total length, as it does not take into account the lower bound value itself. To correct this, the second term $|J(2)| \cdot \sum I(2) = 4 \cdot 2$ subtracts the repetitive contribution of the lower bound 2, once for each possible upper bound. Therefore:*

$$\begin{aligned} TIL_{m_1}(g_1) &= |\{2\}| \times (2 + 3 + 4 + 6) - |\{2, 3, 4, 6\}| \times 2 \\ &= (1 \times 15) - (4 \times 2) = 7 \end{aligned}$$

This leads to the exact sum of lengths of the intervals $[2, 2]$, $[2, 3]$, $[2, 4]$ and $[2, 6]$, i.e., $0 + 1 + 2 + 4 = 7$.

- *Total interval lengths for m_2: The value of g_1 for m_2 is 8. Similarly, $TIL_{m_2}(g_1)$ computes the total length of all intervals that contain the value 8. Here, $I(8) = \{7, 8\}$ and $J(8) = \{8, 9, 12\}$. The term $|I(8)| \cdot \sum J(8) = 2 \cdot (8+9+12)$ sums all upper bounds for each compatible lower bound. This overestimates the total interval lengths. The correction term $|J(8)| \cdot \sum I(8) = 3 \cdot (7+8)$ subtracts the repetitive contribution of the lower bounds. Therefore:*

$$\begin{aligned} TIL_{m_2}(g_1) &= |\{7, 8\}| \times (8 + 9 + 12) - |\{8, 9, 12\}| \times (7 + 8) \\ &= (2 \times 29) - (3 \times 15) = 13 \end{aligned}$$

The resulting value corresponds to the exact sum of lengths of the intervals $[7, 8]$, $[7, 9]$, $[7, 12]$ $[8, 8]$,$[8, 9]$ and $[8, 12]$ i.e., 1+2+5+0+1+4 = 13.

Total Hyper-volumes. Once the total lengths of the intervals have been computed for each attribute $m \in \mathcal{M}$, the product of these interval lengths across all attributes gives the total hyper-volume of the interval patterns covering the object g. which is formally defined as follows:

$$IPH(g) = \prod_{m \in \mathcal{M}} TIL_m(g) \tag{3}$$

The proof of the IPH function is provided in Appendix A

Example 5. *Continuing from the previous example, the sum of the hyper-volumes of all interval patterns covering object g_1 is obtained by multiplying the total interval lengths for attributes m_1 and m_2:*

$$\begin{aligned} IPH(g_1) &= TIL_{m_1}(g_1) \cdot TIL_{m_2}(g_1) \\ &= 7 \times 13 = 91 \end{aligned}$$

5.3. HFIPS Sampling Algorithm

This section describes the HFIPS sampling algorithm (see Algorithm 2). The algorithm starts by computing, for each object g in the database $\mathcal{N}$, the total hyper-volume of interval patterns covering g (line 3), using the `IPH` function (see Section 5.2). Line 4 implements the first step of the two-step sampling procedure, where an object g is drawn with probability proportional to the sum of hyper-volumes of the interval patterns that cover it. In the

second step (from line 6, to line 13), an interval pattern covering at least the selected object g is sampled. The pattern is constructed attribute by attribute. For each attribute $m \in \mathcal{M}$, a lower bound a_m is drawn from the set of candidate distinct values $I(v_{g,m})$, using a weight function w_{lb} that incorporates the cumulative contribution of potential upper bounds (line 9). This bias favors smaller values of a_m, which are more likely to lead to larger intervals. Then, an upper bound b_m is sampled from the set of candidate values $J(v_{g,m})$, using a weight function w_{ub} that promotes wider intervals by taking into account the difference $b_m - a_m$ (line 12). Once both bounds are drawn, the corresponding interval is added to the pattern under construction (line 13).

5.4. Sampling Distribution

In this section, we demonstrate that HFIPS samples interval patterns in proportion to their frequencies $\times$ hyper-volumes.

Property 3. *For a numerical database $\mathcal{N}$, the HFIPS algorithm samples an interval pattern $\mathcal{V}$ with probability proportional to the product of its hyper-volume and frequency. Formally:*

$$P[\mathcal{V}] \propto vol(\mathcal{V}) \cdot freq(\mathcal{V})$$

Proof. Let $\mathcal{L}$ be the set of all interval patterns. We define the normalization constant Z as:

$$Z = \sum_{\mathcal{V} \in \mathcal{L}} vol(\mathcal{V}) \cdot freq(\mathcal{V}) = \sum_{g \in \mathcal{G}} IPH(g).$$

The HFIPS sampling procedure operates in two sequential steps. First, an object $g \in \mathcal{G}$ is sampled with a probability $P(g) = \frac{IPH(g)}{\sum_{g' \in \mathcal{G}} IPH(g')}$. Second, given g from the first step, an interval pattern $\mathcal{V}$ that covers g is drawn with a conditional probability proportional to its weight. We demonstrate first that $P[\mathcal{V} \mid g] = \frac{vol(\mathcal{V})}{IPH(g)}$. Then, we demonstrate that the overall probability of drawing a particular interval pattern $\mathcal{V}$ by HFIPS is equal to $P[\mathcal{V}] = \frac{vol(\mathcal{V})}{Z} \times freq(\mathcal{V})$.

For a given object $g \in \mathcal{G}$ and for each attribute $m \in \mathcal{M}$, HFIPS constructs an interval $[\underline{w_m}, \overline{w_m}]$ by performing two sequential sampling steps. Consider the candidate lower and upper bound sets: $I(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \leq v_{g,m}\}$ and $J(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \geq v_{g,m}\}$.

Algorithm 2: Sampling an interval pattern proportionally to the product of its hyper-volume and frequency (HFIPS)

Input: A numerical database $\mathcal{N}$;
Output: An interval pattern $\mathcal{V}$ sampled proportionally to its hyper-volume $\times$ frequency;

Pre-processing: Compute $w_{HF}(g) = IPH(g)$ for each object $g \in \mathcal{G}$;

Step 1: Draw an object $g \in \mathcal{G}$ proportionally to it weight w_{HF} ;

$\mathcal{V} \longleftarrow \langle\rangle$; `/* Interval pattern under construction */`
Step 2: foreach *attribute* $m \in \mathcal{M}$ **do**

$I(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \leq v_{g,m}\}$; `/* Compute the candidate lower bound set */`

$J(v_{g,m}) = \{v \in \mathcal{N}_m \mid v \geq v_{g,m}\}$; `/* Compute the candidate upper bound set */`

Draw a lower bound $a_m \in I(v_{g,m})$ proportionally to w_{lb}, where:

$$w_{lb}(a_m) = \left(\sum_{x \in J_{g,m}} x\right) - |J_{g,m}| \cdot a_m;$$

Draw an upper bound $b_m \in J(v_{g,m})$ proportionally to w_{ub}, where:

$$w_{ub}(b_m) = b_m - a_m;$$

$\mathcal{V} \longleftarrow \mathcal{V} + +[a_m, b_m]$; `// Adding` $[a_m, b_m]$ `to the interval pattern in construction`

return $\mathcal{V}$

1. **Lower Bound Selection:** Each candidate lower bound $v \in I(v_{g,m})$ is assigned a weight $w_{lb}(v)$. The probability of selecting a specific lower bound $\underline{w_m}$ is given by:

$$P(\underline{w_m} \mid g) = \frac{w_{lb}(\underline{w_m})}{\sum_{v \in I(v_{g,m})} w_{lb}(v)} \quad \text{with}$$

$$w_{lb}(v) = \left(\sum_{x \in J(v_{g,m})} x \right) - |J(v_{g,m})| \cdot v$$

The normalizing constant $\sum_{v \in I(v_{g,m})} w_{lb}(v)$ corresponds to the Total Intervals Length (cf. Equation 2). This allows the normalization constant to be computed directly as:

$$\begin{aligned} \sum_{v \in I(v_{g,m})} w_{lb}(v) &= |I(v_{g,m})| \cdot \left(\sum_{x \in J(v_{g,m})} x \right) - |J(v_{g,m})| \cdot \left(\sum_{v \in I(v_{g,m})} v \right) \\ &= TIL_m(g). \end{aligned}$$

Then:

$$P(\underline{w_m} \mid g) = \frac{w_{lb}(\underline{w_m})}{TIL_m(g)}$$

2. **Upper Bound Selection:** Once the lower bound $\underline{w_m}$ has been sampled, the upper bound $\overline{w_m}$ is selected from the set $J(v_{g,m})$ according to the following probability:

$$P(\overline{w_m} \mid g, \underline{w_m}) = \frac{\overline{w_m} - \underline{w_m}}{w_{lb}(\underline{w_m})}.$$

This formulation favors larger intervals, as the probability increases with the difference $\overline{w_m} - \underline{w_m}$.

Therefore, the overall conditional probability of selecting the interval $[\underline{w_m}, \overline{w_m}]$ for an attribute m is given by:

$$\begin{aligned} P\big([\underline{w_m}, \overline{w_m}] \mid g\big) &= P(\underline{w_m} \mid g) \cdot P(\overline{w_m} \mid g, \underline{w_m}) \\ &= \frac{w_{lb}(\underline{w_m})}{TIL_m(g)} \cdot \frac{\overline{w_m} - \underline{w_m}}{w_{lb}(\underline{w_m})} \\ &= \frac{\overline{w_m} - \underline{w_m}}{TIL_m(g)}. \end{aligned}$$

Since the intervals for different attributes are selected independently, the probability of constructing the entire interval pattern $\mathcal{V} = \langle [\underline{w_m}, \overline{w_m}] \rangle_{m \in \{1,\ldots,|\mathcal{M}|\}}$ covering an object $g \in \mathcal{G}$ is:

$$P[\mathcal{V} \mid g] = \frac{\prod_{[\underline{w_m}, \overline{w_m}] \in \mathcal{V}} (\overline{w_m} - \underline{w_m})}{\prod_{m \in \mathcal{M}} TIL_m(g)}$$

Using the definition of hyper-volume and Equation 3, we have:

$$P[\mathcal{V} \mid g] = \frac{vol(\mathcal{V})}{IPH(g)}$$

By the law of total probability [21], the overall probability of drawing a particular interval pattern $\mathcal{V}$ is

$$P[\mathcal{V}] = \sum_{g \in cover(\mathcal{V})} P[\mathcal{V} \mid g] \cdot P(g).$$

Where $P(g)$ is the probability of sampling g according to Step 1 of Algorithm 2. Substituting the expressions from the two steps results in:

$$P[\mathcal{V}] = \sum_{g \in cover(\mathcal{V})} \frac{vol(\mathcal{V})}{IPH(g)} \cdot \frac{IPH(g)}{\sum_{g' \in \mathcal{G}} IPH(g')} = \frac{vol(\mathcal{V}) \cdot freq(\mathcal{V})}{\sum_{g' \in \mathcal{G}} IPH(g')}.$$

Defining $Z = \sum_{g' \in \mathcal{G}} IPH(g')$, the normalization constant, completes the derivation:

$$P[\mathcal{V}] = \frac{vol(\mathcal{V})}{Z} \times freq(\mathcal{V}).$$

□

5.5. Complexity analysis of HFIPS

We now analyze the time complexity of the HFIPS approach. The analysis is structured into three stages: data pre-processing, object sampling and interval construction.

Property 4. *The overall time complexity of HFIPS for sampling one interval pattern is:*

$$O\Big(|\mathcal{G}| \cdot |\mathcal{M}| + log(|\mathcal{G}|) + |\mathcal{M}| \cdot (|\mathcal{G}| + \log |\mathcal{G}|)\Big).$$

Proof. The numerical database $\mathcal{N}$ contains $|\mathcal{G}|$ objects and $|\mathcal{M}|$ attributes. In the worst case, each attribute has $|\mathcal{G}|$ distinct values. For every object g and attribute m, the candidate sets $I(v_{g,m})$ and $J(v_{g,m})$ are computed using a linear scan over $\mathcal{N}_m$, requiring $O(2 \cdot |\mathcal{G}|)$ time.

The overall sampling procedure can be divided into three phases:

- **Pre-processing:** For each object $g \in \mathcal{G}$, we compute its weight using the function $IPH(g)$. This involves computing the function $TIL_m(g)$, for every attribute $m \in \mathcal{M}$, which takes $O(2 \cdot |\mathcal{G}|)$ time per attribute. Since $IPH(g)$ is the product over all attributes, the total pre-processing time is $O(|\mathcal{M}| \cdot 2 \cdot |\mathcal{G}|)$.

- **Step 1:** An object $g \in \mathcal{G}$ is drawn proportionally to its weight $IPH(g)$. This step is performed using a dichotomic search, which has a time complexity of $O(\log |\mathcal{G}|)$.

- **Step 2:** For each attribute $m \in \mathcal{M}$, the following substeps are performed:

  1. For each lower bound candidate $a_m \in I(v_{g,m})$, compute its weight using w_{lb} in $O(|\mathcal{G}|)$ time.
  2. Draw a lower bound a_m proportionally to w_{lb} using a dichotomic search in $O(\log |\mathcal{G}|)$.
  3. For each upper bound candidate $b_m \in J(v_{g,m})$, compute its weight using w_{ub} in $O(|\mathcal{G}|)$ time.
  4. Draw an upper bound b_m proportionally to w_{ub} using a dichotomic search in $O(\log |\mathcal{G}|)$.

  Hence, Step 2 has an asymptotic total complexity of

$$O\big(|\mathcal{M}| \cdot (|\mathcal{G}| + \log |\mathcal{G}|)\big).$$

By combining all three phases, the asymptotic overall time complexity of the HFIPS sampling procedure is:

$$O\Big(|\mathcal{G}| \cdot |\mathcal{M}| + log(|\mathcal{G}|) + |\mathcal{M}| \cdot (|\mathcal{G}| + \log |\mathcal{G}|)\Big).$$

□

In practice, an analyst may wish to sample k patterns. Repeat HFIPS k times requires $O(k \cdot (log(|\mathcal{G}|) + |\mathcal{M}| \cdot (|\mathcal{G}| + log(|\mathcal{G}|))))$ in addition to the pre-processing time which is in $O(|\mathcal{M}| \cdot |\mathcal{G}|)$. However, this complexity can be reduced by storing the sets $I(v_{g,m})$ and $J(v_{g,m})$ in memory during pre-processing, which avoids recomputing them during sampling.

5.6. Sampling distribution and IPH

The following property quantifies the sampling probability of an object to the hyper-volumes of its interval patterns. We show that an object g is more likely to be sampled than under uniform sampling if and only if the sum of the hyper-volumes of the interval patterns that cover g exceeds the average hyper-volume.

Property 5. *The sampling distribution satisfies:*

$$P(g) \geq \frac{1}{|\mathcal{G}|} \iff IPH(g) \geq \frac{1}{|\mathcal{G}|} \sum_{g' \in \mathcal{G}} IPH(g')$$

Proof. By construction, our algorithm selects an object $g \in \mathcal{G}$ with probability

$$P(g) = \frac{IPH(g)}{\sum_{g' \in \mathcal{G}} IPH(g')}.$$

An object g is therefore sampled with a probability at least as high as under uniform sampling (i.e., $\frac{1}{|\mathcal{G}|}$) if and only if its hyper-volume $IPH(g)$ is at least equal to the average hyper-volume:

$$P(g) \geq \frac{1}{|\mathcal{G}|} \iff IPH(g) \geq \frac{1}{|\mathcal{G}|} \sum_{g' \in \mathcal{G}} IPH(g').$$

Let $A = \sum_{g' \in \mathcal{G}} IPH(g')$ denote the total hyper-volume. Then:

$$P(g) = \frac{IPH(g)}{A}.$$

The inequality $P(g) \geq \frac{1}{|\mathcal{G}|}$ is equivalent to:

$$\frac{IPH(g)}{A} \geq \frac{1}{|\mathcal{G}|} \quad \iff \quad IPH(g) \geq \frac{A}{|\mathcal{G}|}.$$

Since $\frac{A}{|\mathcal{G}|}$ corresponds to the average hyper-volume, this shows that g is more likely to be sampled than under uniform sampling if and only if its hyper-volume exceeds the average. □

6. Experiments and Results

This section experimentally evaluates the performance of both FIPS and HFIPS by addressing the following questions.

1. How reliable is the frequency (resp. hyper-volume times frequency) of interval patterns sampled by FIPS (resp. HFIPS)?
2. How does the long-tail phenomenon affect FIPS and HFIPS?
3. How diverse are the interval patterns sampled by FIPS and HFIPS?
4. How does FIPS and HFIPS perform in term of CPU time?
5. What is the relevance of the patterns sampled by FIPS and HFIPS according to the plausibility criterion?

Selected databases. Our experimental evaluation is conducted on a set of 13 numerical databases. *Glass*, *Iris*, *balance-Scale*, *Diabetes*, *sonar* and *heart* databases are from the UCI Machine Learning Repository[1], while the 7 remaining are from the experimental protocol of our earlier work [22]. The number of numerical attributes, objects and distinct values for each database is presented in Table 2. The results of frequency, plausibility and CPU time experiments are presented for a reduced number of databases. Further results are available in Appendix B and at: https://github.com/djawed-bkh/Interval-Patterns-Sampling

Compared approaches. To the best of our knowledge, there is no approach for sampling interval patterns from numerical data, so we compare FIPS and HFIPS with a first method that performs a uniform sampling of interval patterns. This first method uniformly draws an attribute $m \in \mathcal{M}$ and then uniformly draws the interval bounds from the values of the attribute m. This procedure is repeated until intervals are constructed for all the database attributes. However, as shown in Table 3, this method samples many patterns with empty coverage, making them uninformative for an analyst.

To address this issue, we add a coverage control as described in Algorithm 3 and we call UNIFORM the resulting method. While there are available attributes in $\mathcal{M}^*$ (line 5), UNIFORM uniformly selects and removes an attribute m from $\mathcal{M}^*$ (lines 6-7). Given the current coverage of the partially

[1] https://archive.ics.uci.edu/

Datasets	$\#\mathcal{M}$	$\#\mathcal{G}$	# Distinct values
NT	3	130	67
AP	5	135	674
BK	5	96	313
Cancer	9	116	900
CH	8	209	396
Yacht	7	308	322
Iris	4	150	123
LW	10	189	253
Glass	9	214	939
balance-scale	4	625	20
diabetes	8	768	1254
heart	13	270	384
sonar	60	208	11 256

Table 2: Datasets characteristics

constructed interval pattern $\mathcal{V}$ (i.e. the subset of objects covered by the intervals already sampled in $\mathcal{V}$), a candidate set $\mathcal{N}_m^*$ is defined (line 8). The interval bounds $[a_m, b_m]$ for attribute m are then sampled by requiring that at least one bound appears in an object within the coverage of $\mathcal{V}$, thereby preventing empty coverage (line 9). The new sampled interval is added to the partially constructed interval pattern $\mathcal{V}$ (line 10). Finally, the sampled interval pattern is returned (line 11). The source code of these methods and experimental results are available at `https://github.com/djawed-bkh/Interval-Patterns-Sampling`

6.1. Distribution of Sampled Patterns and Long Tail Phenomenon

Sections 6.1.1 and 6.1.2 address Question 1. We evaluate the patterns sampled by FIPS and HFIPS according to their respective interestingness measures. These patterns are compared to patterns sampled uniformly using UNIFORM. We address Question 2 in Section 6.1.3 by evaluating the robustness of each approach with regard to the long tail phenomenon.

6.1.1. Evaluation of Patterns with Respect to the Frequency

Figure 1 presents the frequencies of 500 patterns sampled by FIPS, HFIPS and UNIFORM. The patterns are sorted in descending order of frequency, and the results are averaged over 10 iterations.

Algorithm 3: UNIFORM: sampling of interval patterns

Input: $\mathcal{N}$: Numerical database;
Output: An interval pattern $\mathcal{V}$ sampled uniformly with a non empty coverage ;
$\mathcal{V} \longleftarrow \langle\rangle$; `// Interval pattern under construction`
$\mathcal{M}^* \longleftarrow \{m_1, ..., m_{|\mathcal{M}|}\}$; `// Remaining attributes`
while $\mathcal{M}^* \neq \emptyset$ **do**
  $m \longleftarrow$ Draw uniformly an attribute from $\mathcal{M}^*$;
  $\mathcal{M}^* \longleftarrow \mathcal{M}^* \backslash \{m\}$;
  $\mathcal{N}_m^* \longleftarrow \{v_{g,m} \mid g \in cover(\mathcal{V})\}$; `// Values from` $\mathcal{N}_m$ `appearing in objects of the current coverage of` $\mathcal{V}$
  $[a_m, b_m] \longleftarrow$ Draw uniformly values a_m, b_m such that:

$$a_m \leq b_m \text{ and } \begin{cases} a_m \in \mathcal{N}_m \wedge b_m \in \mathcal{N}_m^* \\ or \\ a_m \in \mathcal{N}_m^* \wedge b_m \in \mathcal{N}_m \end{cases}$$

  ; `/* selecting from` $\mathcal{N}_m^*$ `ensures the coverage of at least one object */`
  $\mathcal{V} \longleftarrow \mathcal{V} + +[a_m, b_m]$;
return $\mathcal{V}$;

We observe in all databases that FIPS samples patterns with higher frequencies than those obtained with UNIFORM. This can be attributed to Step 1 of FIPS, which is biased towards selecting objects covered by a large number of patterns, leading to higher frequency. HFIPS amplifies this frequency bias by jointly considering hyper-volume with frequency into the sampling procedure.

6.1.2. Evaluation of Patterns with Respect to Hyper-volume × Frequency

Figure 2 shows the values of the product of hyper-volume and frequency of 500 patterns sampled by FIPS, HFIPS and UNIFORM, with patterns sorted in descending order according to this measure. The results are averaged over 10 iterations.

For all databases, FIPS samples patterns with higher values compared to those obtained with UNIFORM. This is explained by the fact that FIPS favors

Databases	Empty coverage (%)
NT	8
AP	76
BK	66
Cancer	98
CH	96
Yacht	81
Iris	56
LW	86
Glass	99
balance-scale	0
diabetes	92
heart	96
sonar	100

Table 3: Percentage of patterns having an empty coverage when sampling interval patterns according to a uniform distribution without coverage control (10 000 sampled patterns).

frequent patterns. Since frequent interval patterns cover more database observations, they are also more likely to have wider intervals, especially when the observations are scattered, resulting in higher volumes and consequently higher values of frequency $\times$ hyper-volume. For all databases, HFIPS samples patterns with higher values compared to those obtained with FIPS. This difference is due to the sampling procedure of HFIPS, which in Step 1 biases the object selection towards those covered by patterns with both high frequencies and hyper-volumes and, in Step 2, biases the selection of interval bounds towards wide intervals. Resulting patterns have high frequencies with large hyper-volumes.

6.1.3. Long Tail Phenomenon

Question 2 related to the long tail phenomenon is addressed in this section. The *long tail phenomenon* is a statistical effect introduced by Bryson [23] and popularized in the context of business by Anderson [24]. It affects sampling techniques by leading to the sampling of uninteresting patterns. The long tail is characterized by an unbalanced distribution of sampled patterns with respect to the considered measure. A small number of patterns with high measure values form the "head", while a large number of patterns with low measure values form the "tail". In the following, we evaluate the robust-

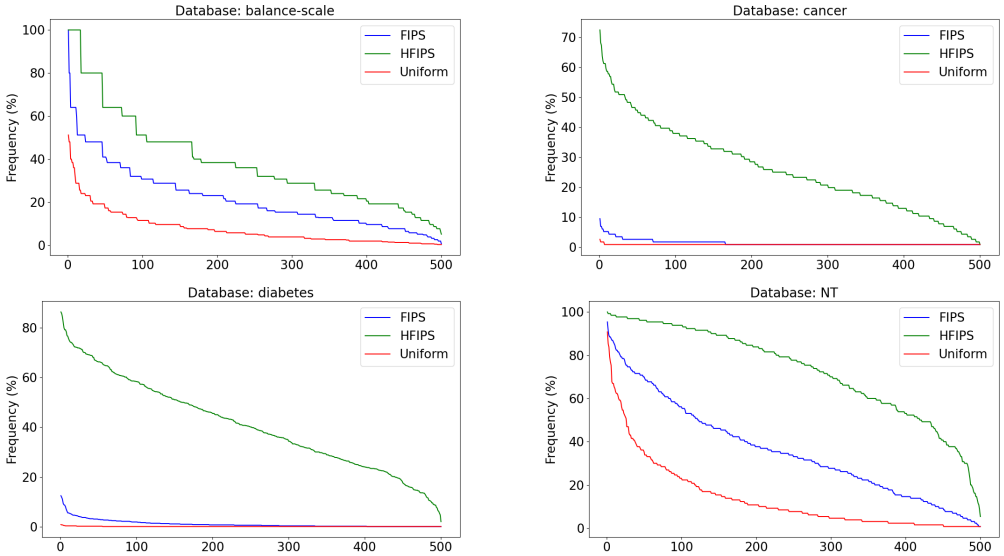


Figure 1: Frequency evaluation for 500 patterns sampled with FIPS and the UNIFORM methods

ness of FIPS and HFIPS to the long tail phenomenon, each one according to its respective interestingness measure, namely frequency for FIPS and the product of frequency and hyper-volume for HFIPS.

Robustness of FIPS to the long tail phenomenon. Figure 1 shows that FIPS is significantly less affected by the long-tail phenomenon than UNIFORM. For the *diabetes* and *cancer* databases, 65% and 68.2% of patterns generated by FIPS occur in the tail (frequency below 1%), in comparison to approximately 99% with UNIFORM. For the *balance-scale* and *NT* databases, FIPS produces a negligible number of low-frequency patterns, making the tail almost undetectable. In contrast, UNIFORM produces 9.4% and 10% of patterns with frequencies below 1% for the balance-scale and NT databases, respectively.

Robustness of HFIPS to the long tail phenomenon Similarly to FIPS, Figure 2 shows that HFIPS is significantly less affected by the long-tail phenomenon than UNIFORM, with respect to the considered interestingness measure. In the *NT* database, the tail is almost undetectable. In the *balance-scale*, *cancer*, and *diabetes* datasets, we observe that HFIPS consistently delays the appearance of the tail. For instance, in *balance-scale*, only 50% of

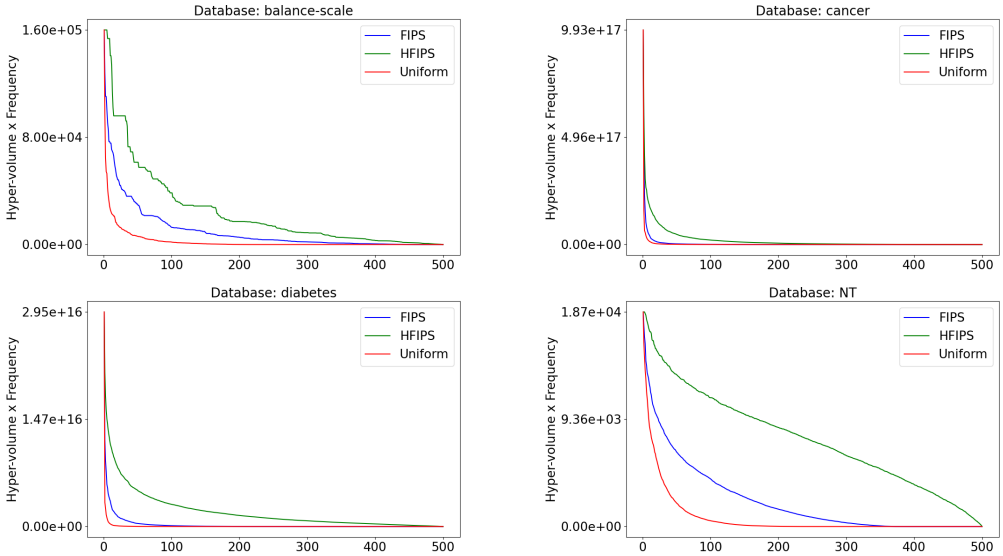


Figure 2: Volume evaluation for 500 patterns sampled with HFIPS and the UNIFORM methods

the sampled patterns fall within the tail, compared to 86% for UNIFORM. In *cancer*, the difference is even more pronounced: 85% for HFIPS versus 99% for UNIFORM. Similarly, in *diabetes*, 60% of the HFIPS patterns appear in the tail, compared to 99% for UNIFORM.

6.2. Diversity Evaluation

Sections 6.2.1 and 6.2.2 address Question 3 on the diversity of sampled patterns. Sampling diverse patterns is essential in an interactive mining process, as it allows the analyst to explore multiple areas of the solution space. In this section, we use various metrics to assess the diversity of patterns sampled by FIPS, HFIPS and UNIFORM. Section 6.2.1 deals with the diversity of equivalence classes among the sampled patterns, while Section 6.2.2 focuses on their overlap in terms of covered objects.

6.2.1. Equivalence Classes Diversity

To evaluate the diversity of FIPS, HFIPS and UNIFORM in terms of equivalence classes, we use the measure presented in [10], which is defined as follows:

$$diversity(K, \mathcal{N}) = \frac{|\{cover(\mathcal{V}_1, \mathcal{N}), \ldots, cover(\mathcal{V}_{|K|}, \mathcal{N})\}|}{|K|}$$

where $|K|$ is the number of sampled patterns. Figure 3 shows that patterns sampled with HFIPS and FIPS have a greater diversity in equivalence classes than those sampled with UNIFORM, except in the case of the *balance-scale* database. This can be attributed to the higher frequency of patterns produced by HFIPS and FIPS (see Section 6.1), which results in wider coverage and, consequently, greater variation in the covered objects across samples, thereby increasing diversity in terms of equivalence classes. The exception observed for the *balance-scale* dataset can be explained by its specific characteristics, in particular the limited number of distinct values and numerical attributes (see Table 2). Since FIPS and HFIPS are biased towards high frequency and high hyper-volume patterns, they tend to sample the same objects repeatedly, resulting in similar intervals and reduced diversity. The higher diversity of HFIPS in comparison to FIPS can be explained by the higher frequencies of patterns sampled by HFIPS, resulting in greater variation in the coverages.

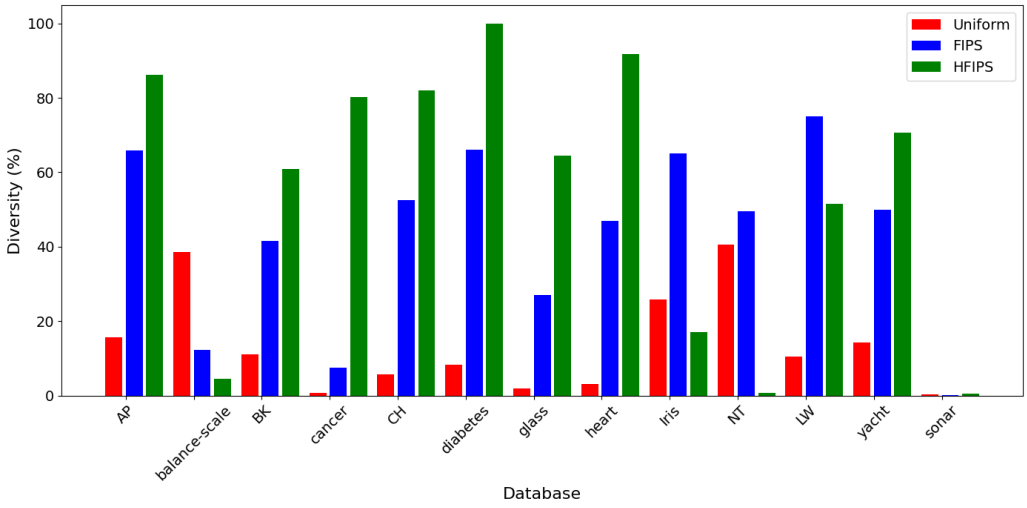


Figure 3: Diversity evaluation of FIPS, HFIPS and UNIFORM methods

6.2.2. Coverage Overlapping

Since the metric presented in Section 6.2.1 does not consider the overlap of coverages between the sampled patterns, we complement the diversity evaluation with a Jaccard index-based measure. We evaluate the coverage diversity of 500 patterns obtained with FIPS, HFIPS and UNIFORM, using the

Jaccard index. As this metric compares pairs of patterns, it cannot be applied directly to a large set. Therefore, we use cumulative distribution functions (CDF) to visually represent the distribution of Jaccard indices computed for all pairs formed by our 500 patterns. Consider a set of $k = 500$ interval patterns $\boldsymbol{\mathcal{V}} = \{\mathcal{V}_1, \ldots, \mathcal{V}_{500}\}$ and a numerical database $\mathcal{N}$.

$$CDF(\mathcal{N}, \boldsymbol{\mathcal{V}}, \theta) = \frac{2 \times |\{(i,j) : Jac(\mathcal{V}_i, \mathcal{V}_j) \leq \theta, \quad 1 \leq i < j \leq k, \quad \mathcal{V}_i, \mathcal{V}_j \in \boldsymbol{\mathcal{V}}\}|}{k(k-1)}$$

Figure 4 shows that, for all databases, the cumulative distribution function curve of HFIPS has a lower area compared to UNIFORM. This indicates that HFIPS generates patterns with higher similarity in terms of coverage than UNIFORM. For small Jaccard indices, HFIPS returns a small number of patterns with low coverage similarity. However, as the similarity threshold increases, the number of patterns produced by HFIPS also grows, meaning that this method generates patterns that share the same objects in their coverage. In contrast, UNIFORM shows greater coverage diversity: a large number of patterns have low Jaccard indices, and the curve generally remains linear as the similarity threshold increases. This means that there is no significant addition of highly similar patterns. This observation can be explained by the fact that UNIFORM tends to sample patterns covering a small number of objects, reducing the probability that these patterns share objects with others from the same approach. In contrast, HFIPS, which samples patterns covering a large number of objects, increases the probability of sampling other patterns that share a subset of objects.

For the FIPS method, we observe that for the *NT* and *balance-scale* databases, the cumulative distribution function curve has a lower area compared to the random sampling approach. This indicates that FIPS generates patterns with higher similarity in terms of coverage than UNIFORM. This can be explained by two factors: first, FIPS samples patterns with greater frequencies than UNIFORM (see Figure 6.1), meaning that patterns obtained with FIPS are more likely to share the same objects in their coverages, leading to lower diversity. The second factor concerns the database characteristics (see Table 2). We observe that *NT* and *balance-scale* have a small number of distinct values, which leads FIPS to often sample the same patterns. Finally, for the *diabetes* and *cancer* databases, we observe that the diversity of FIPS is almost as high as that of UNIFORM. This can be explained by the large number of distinct values in these databases, which leads FIPS to sample more diversified patterns.

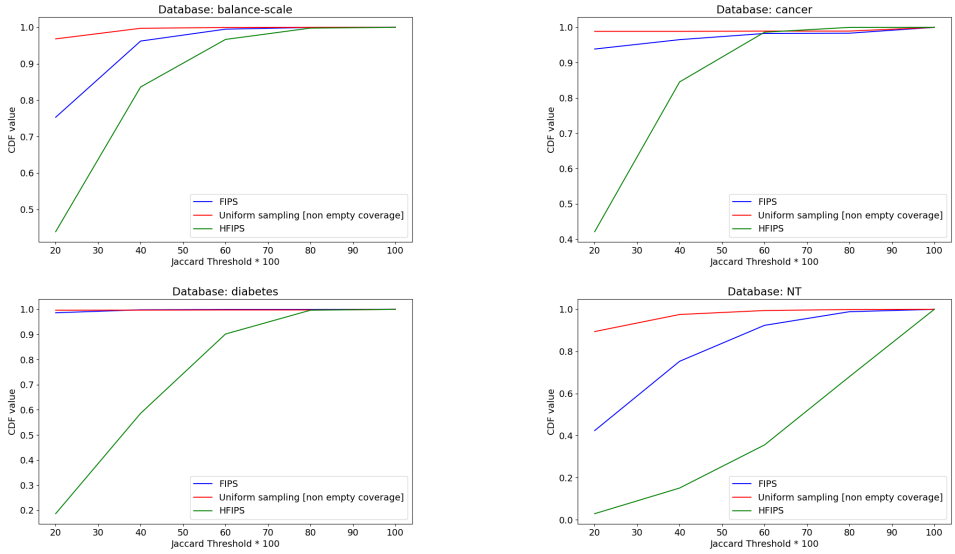


Figure 4: Coverage diversity evaluation for 500 patterns sampled with FIPS, HFIPS and the UNIFORM method

6.3. CPU Time Evaluation

This section deals with running time (Question 4). Figure 5 presents the CPU time[2] required to sample each of 500 interval patterns using FIPS, HFIPS, and UNIFORM. The results are averaged over 10 iterations.

We observe that for most of the databases, FIPS is faster than both UNIFORM and HFIPS. This can be explained by the fact that UNIFORM requires, for each sampled interval in the pattern, a search for distinct values from the current coverage. This is achieved by recalculating the current coverage of the pattern being constructed for each newly sampled interval ($|\mathcal{M}|$ times in total). This procedure also explains the oscillations observed for this method. If the gap between the bounds of the first sampled intervals is large, the coverage computation time will be high. In contrast, if the gap is small, the computation time will be lower. With the exception of *balance-scale*, this time is longer for HFIPS than for UNIFORM and FIPS. This is due to the second step of its algorithm (see Algorithm 2), where the weight computation of the values that are candidates for being bound takes more CPU time. The exception on *balance-scale* can be explained by the small number of distinct

[2]Configuration: Intel Core i7, 11th generation, 3 GHz, 8 cores, 30 GB RAM

values present in this database (see Table 2), which results in a smaller number of candidate values for being bounds, thus requiring less computation time for the second step of HFIPS (see Algorithm 2). Note that in all cases, run-times remain very short as the time scale is measured in milliseconds.

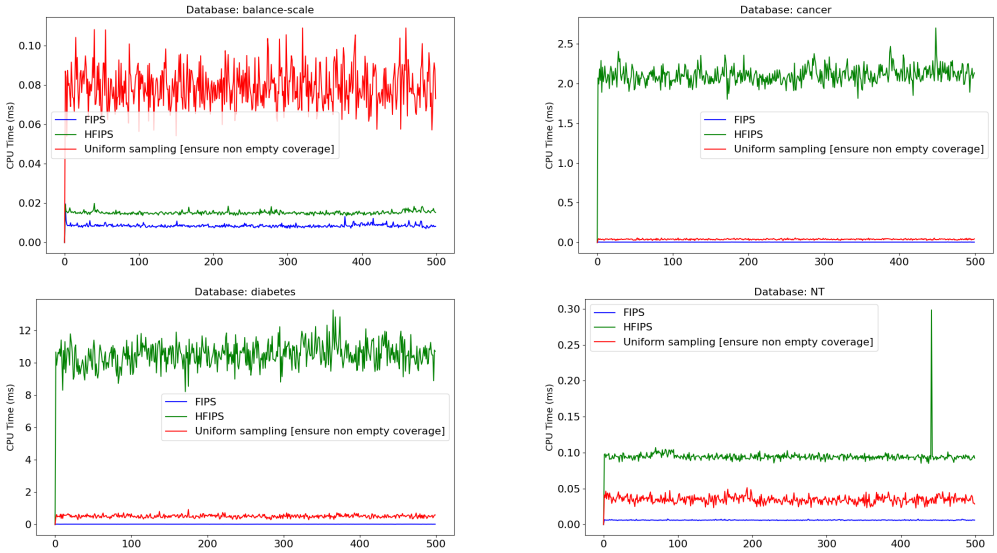


Figure 5: CPU time evolution for a set of 500 patterns sampled by the FIPS, HFIPS and UNIFORM methods

6.4. Plausibility Evaluation

This section addresses Question 5 on the relevance of the patterns sampled by FIPS, HFIPS and UNIFORM according to the plausibility criterion. We use the protocol introduced by [25] and adapted to numerical data in [10]. The idea is to evaluate the relevance of a pattern by comparing its frequency in the original database with its frequency in a randomized database, denoted $\mathcal{N}_{rand}$, in which correlations are broken. This randomization is performed by repeatedly selecting at random, from the original database, two values of the same attribute from two different objects and swapping them. This process is applied independently for each attribute, and a random number of swaps is performed. A pattern's relevance increases with the difference between its frequency in the original database and in randomized variants. Formally,

plausibility is defined as :

$$Plausibility(K, \mathcal{N}) = \frac{\sum_{\mathcal{V}_i \in K} \sum_{j=1}^{R} (freq(\mathcal{V}_i,\ \mathcal{N}) - freq(\mathcal{V}_i,\ \mathcal{N}^j_{rand}))}{\sum_{i=1}^{|K|} (R \times freq(\mathcal{V}_i, \mathcal{N}))}$$

where R is the number of randomised databases. We defined frequency intervals (see Figure 6) and for each interval and method we sampled 10,000 patterns, rejecting those whose frequencies did not fall within the specified range. The restriction imposed by the frequency intervals aims to simulate the interest of an analyst who rejects both highly frequent patterns (often too general) and very rare patterns (often not representative). Considering multiple intervals allow us to simulate several thresholds values considered useful. Sampling was conducted within a time limit of 5 minutes.

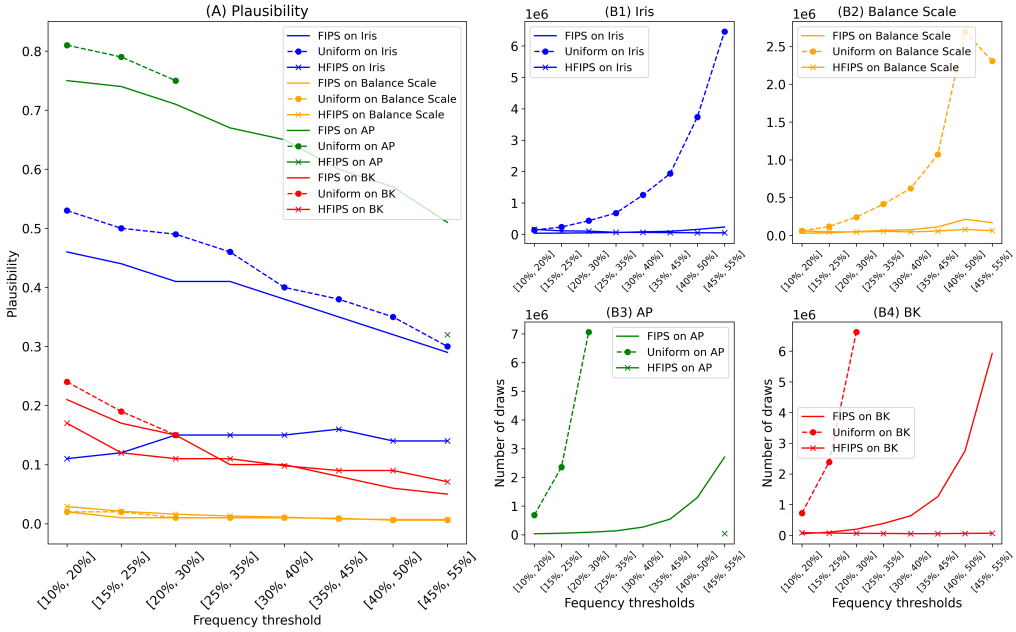


Figure 6: Plausibility evaluation for the FIPS, HFIPS and UNIFORM methods (10 000 sampled patterns) on multiple frequency thresholds

Plausibility. Plausibility tends to decrease as frequency increases, as the more frequent a pattern is, the more likely it is to appear in a randomized database. Since the FIPS and HFIPS sampling are biased towards frequency

and the product of hyper-volume and frequency, respectively, patterns sampled by these methods are expected to have lower plausibility on average than those sampled by UNIFORM. Part A of Figure 6 confirms this observation. As UNIFORM produces patterns with the lowest frequencies, their plausibility is the highest. Conversely, HFIPS, which tends to sample the most frequent patterns, leads to patterns with the lowest plausibility. The results also show that as the frequency threshold increases, the plausibility of the patterns sampled by the different methods converges. Note that some methods fail to sample the required number of patterns within the allocated time. For instance, UNIFORM fails to sample the required number of patterns with frequencies greater than 30% for the *AP* and *BK* databases. On the other hand, HFIPS fails to sample the required number of patterns in *AP* with a frequency lower than 45% within the allocated time.

Number of draws evaluation. Parts B_1, B_2, B_3 and B_4 in Figure 6 show that across all databases, the number of draws required by UNIFORM to reach the target number of patterns is significantly higher than that of FIPS and HFIPS. For frequency thresholds between 10% and 35%, UNIFORM requires 0.87 to 10 times more draws than HFIPS on *iris*, 1.04 to 7 times more draws on *balance-scale*, and 2 to 11 times more draws than FIPS on both *iris* and *balance-scale*. This gap increases for higher frequency thresholds (35% to 45%), reaching up to 27 times more draws than FIPS and 35 times more draws than HFIPS on the *iris* database. For FIPS, the difference is even more pronounced on the *AP* and *BK* databases, where UNIFORM requires 15 to 80 times more draws in low frequency thresholds. At higher thresholds, UNIFORM fails to sample the required number of patterns in the allocated time. Similarly, for HFIPS, the gap is particularly pronounced on the *BK* and *iris* databases, where UNIFORM requires up to 100 times more draws, and reaches 134 times more on *iris*.

7. Conclusion and Perspectives

In this paper, we presented two methods for sampling patterns from numerical data. These methods rely on an interval-based representation of the numerical data that preserves all the original information. The first method, FIPS, samples patterns proportionally to the frequency interestingness measure. HFIPS incorporates the hyper-volume measure into the sampling process in order to sample interval patterns proportionally to the product of frequency and hyper-volume. We theoretically proved that both

FIPS and HFIPS sample patterns proportionally to their respective interestingness measures. We experimentally evaluated the quality of the sampled patterns using several criteria, including frequency, hyper-volume, diversity, efficiency, and plausibility. We also evaluate the robustness of our approaches to the long-tail phenomenon.

Increasingly, analysts want to interact with mining systems by specifying, through queries, the types of patterns they want to explore. In this context, a future work is to design a framework for sampling interval patterns that satisfy user preferences and follows the desired distribution. This task is particularly challenging since, unlike itemsets, interval patterns are defined across all attributes of the database and include hidden dependencies between intervals. Another direction is to tackle other interestingness measures that are relevant to numerical data, such as density. Finally, exploring the use of sampling in machine learning contexts could represent a promising direction. For example, sampling approaches could help identify meaningful hyper-parameters for supervised learning tasks. By guiding the search toward informative regions of the hyper-parameter space, our approaches may help to reduce computational costs and improve the explainability of the resulting models, which would be a step toward more interpretable machine learning models.

Acknowledgement. This work was supported by the French National Research Agency (ANR) and Region Normandie under grant HAISCoDe, and partly funded by ANR under project ANR-24-CE23-0950.

Appendix A. Proof of the IPH function

Proof. We aim to prove that the IPH function correctly computes the sum of the hyper-volumes of all interval patterns that cover a given object $g \in \mathcal{G}$. This requires demonstrating that, for any attribute $m \in \mathcal{M}$, the inner term of Equation 3 corresponds to the sum of the lengths of all intervals that contain the value $v_{g,m}$ of g on attribute m.

Assume that the distinct values of the m attribute are given in a sorted order by:

$$\mathcal{D}_m = \{d_1, d_2, \ldots, d_k\}, \quad \text{with} \quad d_1 < d_2 < \cdots < d_k.$$

Let p be the index of the highest value in $\mathcal{D}_m$ being less than or equal to $v_{g,m}$:

$$p = \max\{i \mid d_i \leq v_{g,m}\}.$$

Then the candidate sets of lower and upper bounds for an attribute m can be defined as follows:

$$I(v_{g,m}) = \{d_1, \ldots, d_p\}, \qquad J(v_{g,m}) = \{d_p, \ldots, d_k\}.$$

The total interval length $TIL(v_{g,m})$ corresponds to the sum of the lengths of all intervals $[a, b]$ such that $a \in I_{g,m}$, $b \in J_{g,m}$, and $a \leq v_{g,m} \leq b$. Formally, we write:

$$TIL_m(g) = \sum_{a \in I_{g,m}} \sum_{b \in J_{g,m}} (b - a).$$

By substituting with the ordered sets, we have:

$$TIL_m(g) = \sum_{i=1}^{p} \sum_{j=p}^{k} (d_j - d_i).$$

By expanding this double summation, we obtain:

$$TIL_m(g) = \sum_{i=1}^{p} \sum_{j=p}^{k} (d_j - d_i) = \sum_{i=1}^{p} \sum_{j=p}^{k} d_j - \sum_{i=1}^{p} \sum_{j=p}^{k} d_i.$$

Since the inner sum is independent of i and repeated p times, the first term can be factored as:

$$\sum_{i=1}^{p} \sum_{j=p}^{k} d_j = p \cdot \sum_{j=p}^{k} d_j,$$

Similarly, since the inner sum is independent of j and repeated $k - p + 1$ times, the second term becomes:

$$\sum_{i=1}^{p} \sum_{j=p}^{k} d_i = (k - p + 1) \cdot \sum_{i=1}^{p} d_i,$$

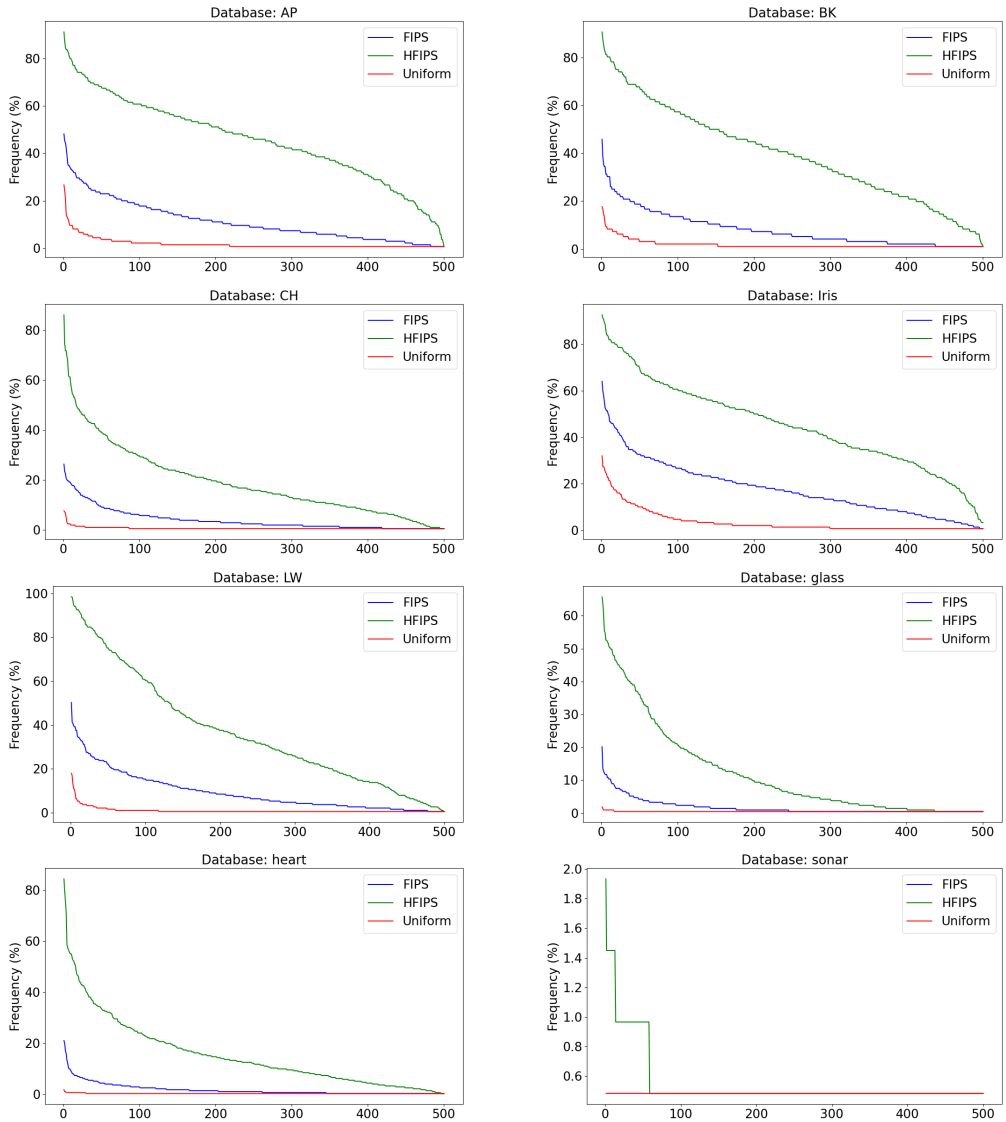
Recognizing that $p = |I(v_{g,m})|$ and $k - p + 1 = |J(v_{g,m})|$, we can rewrite this as:

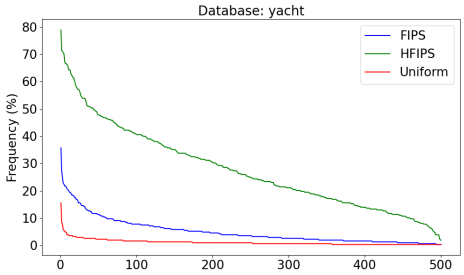
$$TIL_m(g) = |I(v_{g,m})| \cdot \sum_{x \in J(v_{g,m})} x - |J(v_{g,m})| \cdot \sum_{x \in I(v_{g,m})} x.$$

which matches the expression given in Equation 3 for an attribute m. Since the hyper-volume of an interval pattern is the product of the lengths of its intervals, applying the product over all attributes leads to the total hyper-volume of all interval patterns covering object g. □

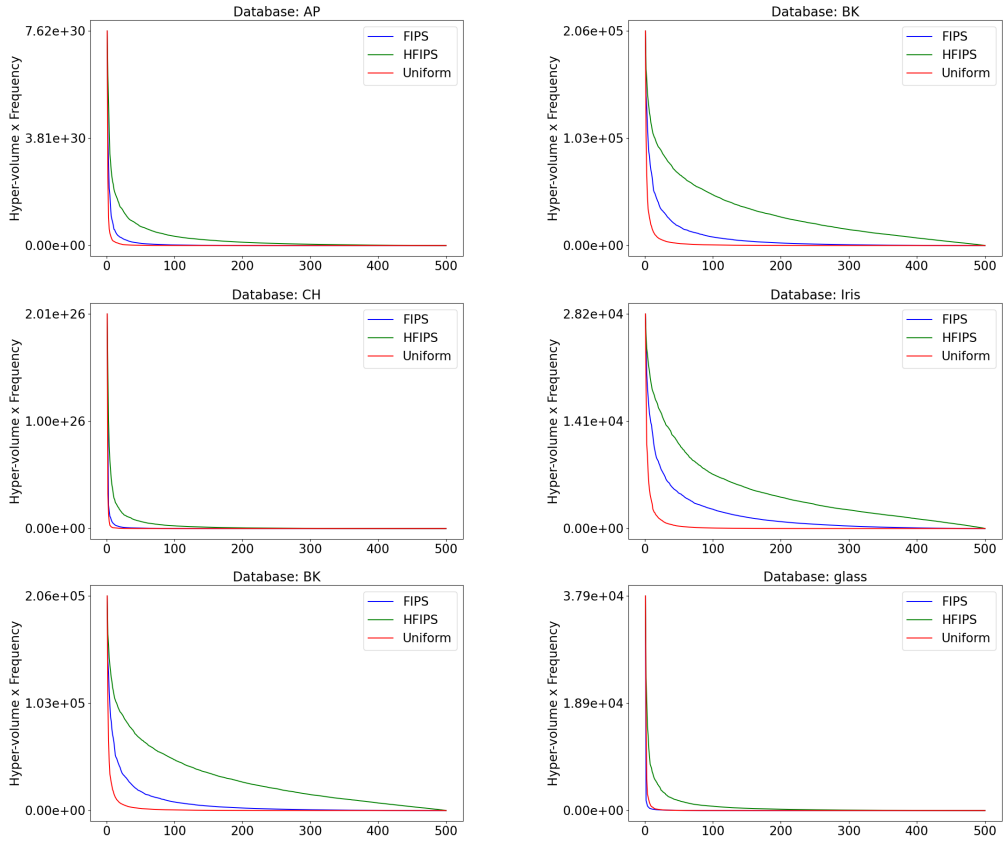
Appendix B. Experimental results

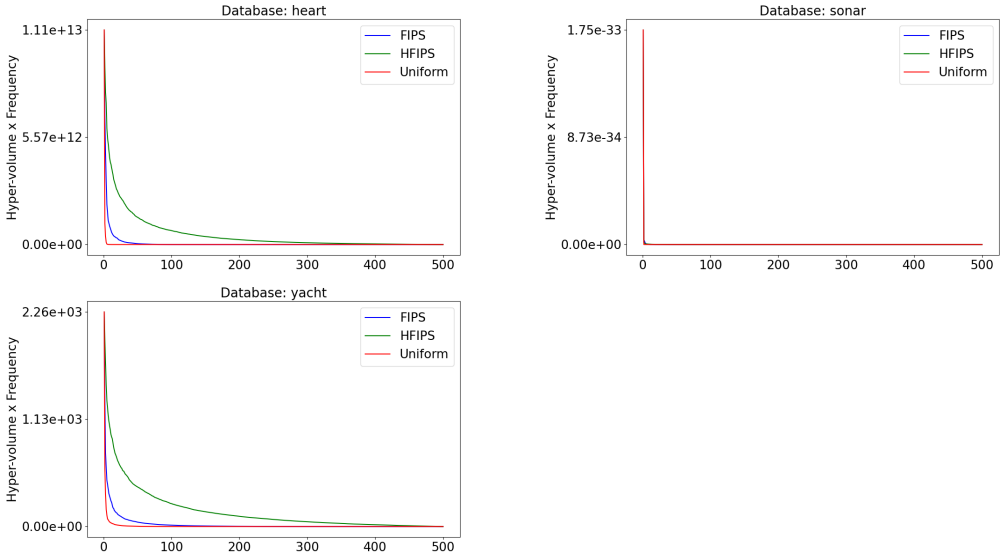
Appendix B.1. Pattern Frequencies Evaluation



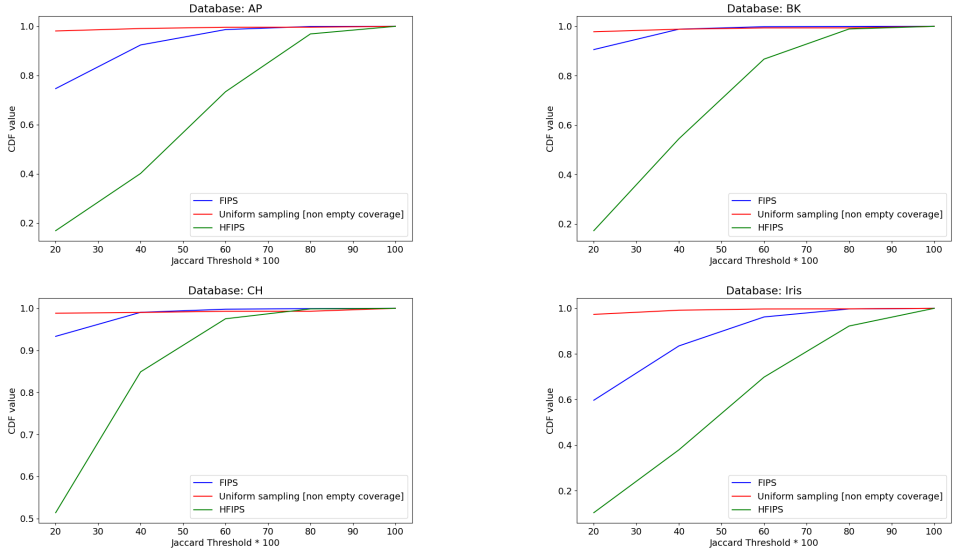


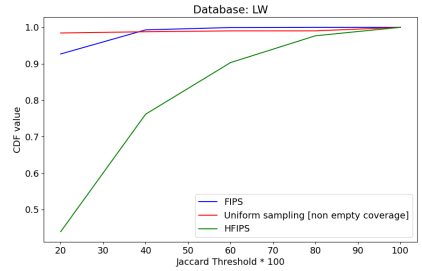
Appendix B.2. Pattern Volume Times Frequency Evaluation

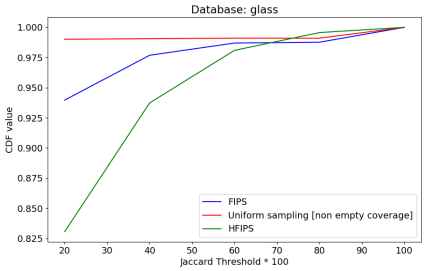


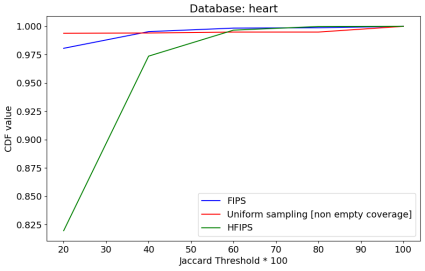


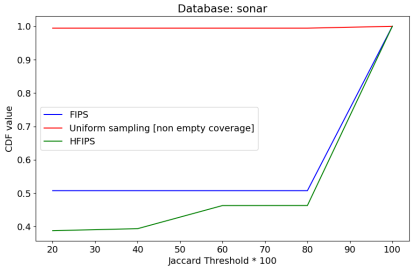
Appendix B.3. Jaccard based diversity evaluation

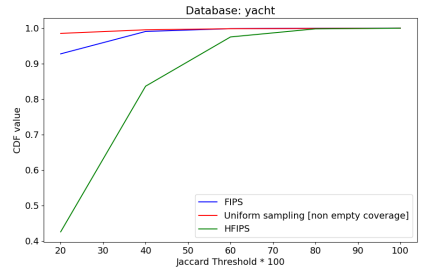




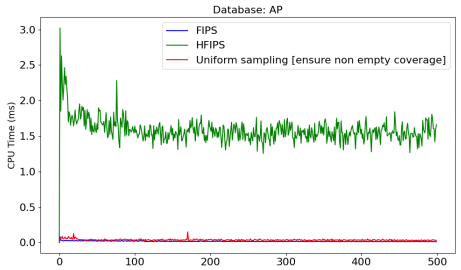


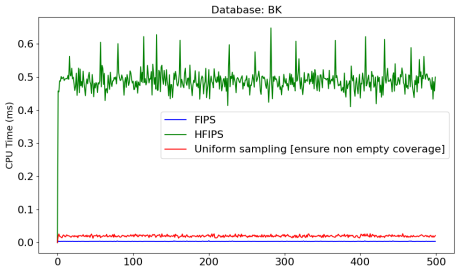


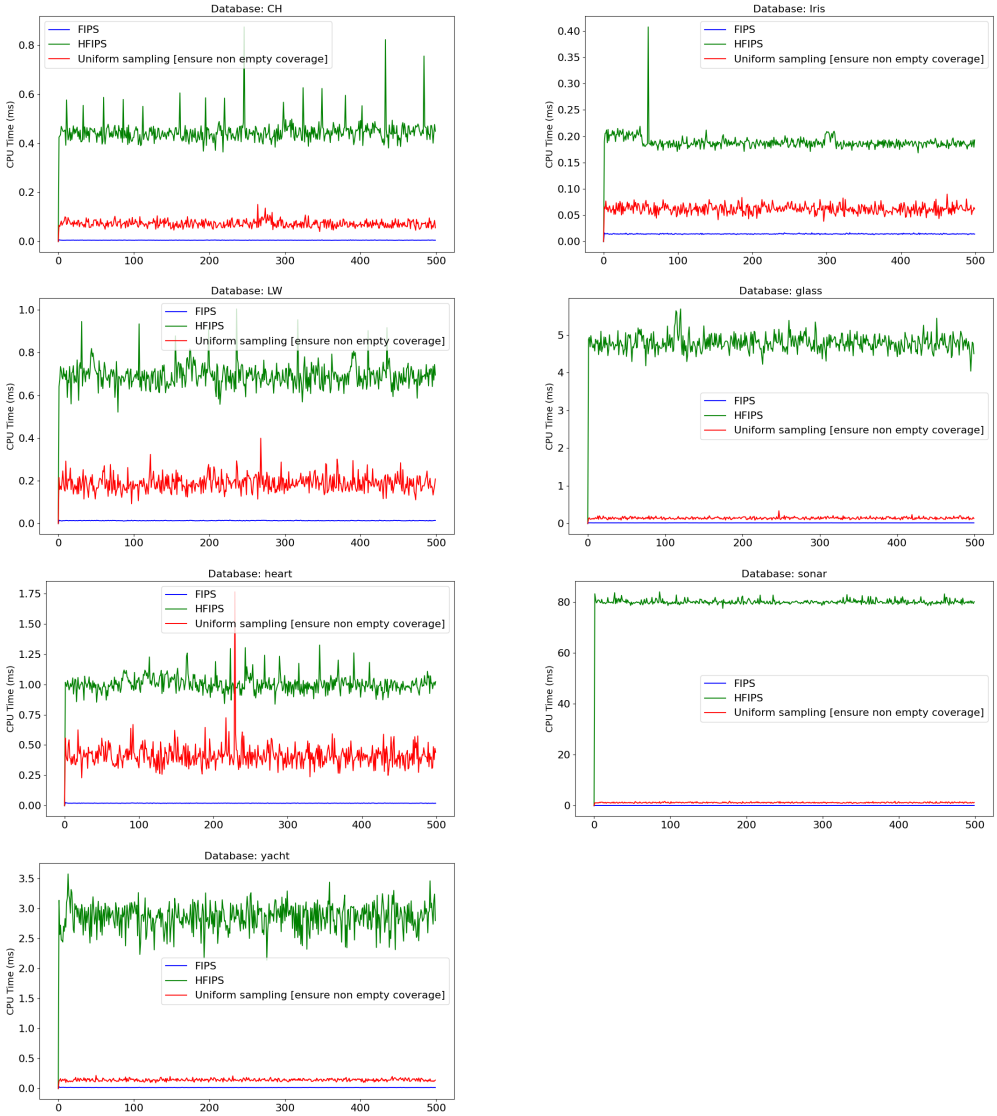




Appendix B.4. CPU time evaluation







References

[1] V. Dzyuba, M. van Leeuwen, Learning what matters - sampling interesting patterns, in: J. Kim, K. Shim, L. Cao, J. Lee, X. Lin, Y. Moon (Eds.), Advances in Knowledge Discovery and Data Mining - 21st Pacific-Asia Conference, PAKDD 2017, Jeju, South Korea, May 23-26, 2017, Proceedings, Part I, Vol. 10234 of Lecture Notes in Com-

puter Science, 2017, pp. 534–546. doi:10.1007/978-3-319-57454-7_42.
URL https://doi.org/10.1007/978-3-319-57454-7_42

[2] M. A. Hasan, M. J. Zaki, Output space sampling for graph patterns, Proc. VLDB Endow. 2 (1) (2009) 730–741. doi:10.14778/1687627.1687710.
URL http://www.vldb.org/pvldb/vol2/vldb09-713.pdf

[3] V. Dzyuba, M. van Leeuwen, L. D. Raedt, Flexible constrained sampling with guarantees for pattern mining, Data Min. Knowl. Discov. 31 (5) (2017) 1266–1293. doi:10.1007/S10618-017-0501-6.
URL https://doi.org/10.1007/s10618-017-0501-6

[4] M. Boley, C. Lucchese, D. Paurat, T. Gärtner, Direct local pattern sampling by efficient two-step random procedures, in: C. Apté, J. Ghosh, P. Smyth (Eds.), Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011, ACM, 2011, pp. 582–590. doi:10.1145/2020408.2020500.
URL https://doi.org/10.1145/2020408.2020500

[5] S. Blachon, R. G. Pensa, J. Besson, C. Robardet, J. Boulicaut, O. Gandrillon, Clustering formal concepts to discover biologically relevant knowledge from gene expression data, Silico Biol. 7 (4-5) (2007) 467–483.
URL http://content.iospress.com/articles/in-silico-biology/isb00321

[6] L. Tschora, T. Guns, E. Pierre, M. Plantevit, C. Robardet, Electricity price forecasting based on order books: a differentiable optimization approach, in: 10th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2023, Thessaloniki, Greece, October 9-13, 2023, IEEE, 2023, pp. 1–10. doi:10.1109/DSAA60987.2023.10302542.
URL https://doi.org/10.1109/DSAA60987.2023.10302542

[7] R. Srikant, R. Agrawal, Mining quantitative association rules in large relational tables, in: SIGMOD 1996 , Canada,, 1996.

[8] J. Dougherty, R. Kohavi, M. Sahami, Supervised and unsupervised discretization of continuous features, in: A. Prieditis, S. Russell (Eds.),

Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995, Morgan Kaufmann, 1995, pp. 194–202. doi:10.1016/B978-1-55860-377-6.50032-3.
URL https://doi.org/10.1016/b978-1-55860-377-6.50032-3

[9] M. Kaytoue, S. O. Kuznetsov, A. Napoli, Revisiting numerical pattern mining with formal concept analysis, in: T. Walsh (Ed.), IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, IJCAI/AAAI, 2011, pp. 1342–1347. doi:10.5591/978-1-57735-516-8/IJCAI11-227.
URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-227

[10] A. Giacometti, A. Soulet, Dense neighborhood pattern sampling in numerical data, in: M. Ester, D. Pedreschi (Eds.), Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018, May 3-5, 2018, San Diego Marriott Mission Valley, San Diego, CA, USA, SIAM, 2018, pp. 756–764. doi:10.1137/1.9781611975321.85.
URL https://doi.org/10.1137/1.9781611975321.85

[11] M. Boley, T. Gärtner, H. Grosskreutz, Formal concept sampling for counting and threshold-free local pattern mining, in: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA, SIAM, 2010, pp. 177–188. doi:10.1137/1.9781611972801.16.
URL https://doi.org/10.1137/1.9781611972801.16

[12] A. Bendimerad, J. Lijffijt, M. Plantevit, C. Robardet, T. D. Bie, Gibbs sampling subjectively interesting tiles, in: M. R. Berthold, A. Feelders, G. Krempl (Eds.), Advances in Intelligent Data Analysis XVIII - 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27-29, 2020, Proceedings, Vol. 12080 of Lecture Notes in Computer Science, Springer, 2020, pp. 80–92. doi:10.1007/978-3-030-44584-3_7.
URL https://doi.org/10.1007/978-3-030-44584-3_7

[13] F. Camelin, S. Loudni, G. Pesant, C. Truchet, Coupling MDL and markov chain monte carlo to sample diverse pattern sets, Data Knowl. Eng. 156 (2025) 102393. doi:10.1016/J.DATAK.2024.102393.
URL https://doi.org/10.1016/j.datak.2024.102393

[14] T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets, in: R. J. B. Jr., B. Goethals, M. J. Zaki (Eds.), FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004, Vol. 126 of CEUR Workshop Proceedings, CEUR-WS.org, 2004.
URL https://ceur-ws.org/Vol-126/uno.pdf

[15] T. M. Opran, S. Loudni, Échantillonnage actif pour la découverte de règles classification via des comparaisons par paires E-41 (2025) 231–238.
URL http://editions-rnti.fr/?inprocid=1002998

[16] L. Diop, C. T. Diop, A. Giacometti, D. Li, A. Soulet, Sequential pattern sampling with norm-based utility, Knowl. Inf. Syst. 62 (5) (2020) 2029–2065. doi:10.1007/S10115-019-01417-3.
URL https://doi.org/10.1007/s10115-019-01417-3

[17] A. Soulet, Echantillonnage de motifs avec une contrainte de fréquence, in: C. Faron, S. Loudcher (Eds.), Extraction et Gestion des Connaissances, EGC 2023, Lyon, France, 16 - 20 janvier 2023, Vol. E-39 of RNTI, Editions RNTI, 2023, pp. 115–126.
URL https://editions-rnti.fr/?inprocid=1002815

[18] L. Diop, High average-utility itemset sampling under length constraints, in: J. Gama, T. Li, Y. Yu, E. Chen, Y. Zheng, F. Teng (Eds.), Advances in Knowledge Discovery and Data Mining - 26th Pacific-Asia Conference, PAKDD 2022, Chengdu, China, May 16-19, 2022, Proceedings, Part II, Vol. 13281 of Lecture Notes in Computer Science, Springer, 2022, pp. 134–148. doi:10.1007/978-3-031-05936-0_11.
URL https://doi.org/10.1007/978-3-031-05936-0_11

[19] L. Diop, M. Plantevit, Scalable sampling for high utility patterns, in: W. Ding, C. Lu, F. Wang, L. Di, K. Wu, J. Huan, R. Nambiar, J. Li, F. Ilievski, R. Baeza-Yates, X. Hu (Eds.), IEEE International Conference on Big Data, BigData 2024, Washington, DC, USA, December 15-18, 2024, IEEE, 2024, pp. 104–109. doi:10.1109/BIGDATA62323.2024.10826031.
URL https://doi.org/10.1109/BigData62323.2024.10826031

[20] S. Moens, M. Boley, Instant exceptional model mining using weighted controlled pattern sampling, in: H. Blockeel, M. van Leeuwen, V. Vinciotti (Eds.), Advances in Intelligent Data Analysis XIII - 13th International Symposium, IDA 2014, Leuven, Belgium, October 30 - November 1, 2014. Proceedings, Vol. 8819 of Lecture Notes in Computer Science, Springer, 2014, pp. 203–214. doi:10.1007/978-3-319-12571-8_18.
URL https://doi.org/10.1007/978-3-319-12571-8_18

[21] W. Feller, An introduction to probability theory and its applications, Volume 2, Vol. 2, John Wiley & Sons, 1991.

[22] D. Bekkoucha, A. Ouali, P. Boizumault, B. Crémilleux, Efficiently mining closed interval patterns with constraint programming, in: B. Dilkina (Ed.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research - 21st International Conference, CPAIOR 2024, Uppsala, Sweden, May 28-31, 2024, Proceedings, Part I, Vol. 14742 of Lecture Notes in Computer Science, Springer, 2024, pp. 51–67. doi:10.1007/978-3-031-60597-0_4.
URL https://doi.org/10.1007/978-3-031-60597-0_4

[23] M. C. Bryson, Heavy-tailed distributions: Properties and tests, Technometrics 16 (1) (1974) 61–68. arXiv:https://www.tandfonline.com/doi/pdf/10.1080/00401706.1974.10489150, doi:10.1080/00401706.1974.10489150.
URL https://www.tandfonline.com/doi/abs/10.1080/00401706.1974.10489150

[24] C. Anderson, The Long Tail: Why the Future of Business Is Selling Less of More, Hyperion, 2006.

[25] A. Gionis, H. Mannila, T. Mielikäinen, P. Tsaparas, Assessing data mining results via swap randomization, in: T. Eliassi-Rad, L. H. Ungar, M. Craven, D. Gunopulos (Eds.), Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006, ACM, 2006, pp. 167–176. doi:10.1145/1150402.1150424.
URL https://doi.org/10.1145/1150402.1150424