Intelligent Data Analysis 0 (0) 1
IOS Press

Floating-point histograms for exploratory analysis of large scale real-world data sets

Marc Boullé
Orange Labs, Lannion, France
E-mail: marc.boulle@orange.com

Abstract. Histograms are among the most popular methods used in exploratory analysis to summarize univariate distributions. In particular, irregular histograms are good non-parametric density estimators that require very few parameters: the number of bins with their lengths and frequencies. Although many approaches have been proposed in the literature to infer these parameters, most existing histogram methods are difficult to exploit for exploratory analysis in the case of real-world data sets, with scalability issues, truncated data, outliers or heavy-tailed distributions. In this paper, we focus on the G-Enum histogram method, which exploits the Minimum Description Length (MDL) principle to build histograms without any user parameter. We then propose to extend this method by exploiting a new modeling space based on floating-point representation, with the objective of building histograms resistant to outliers or heavy-tailed distributions. We also suggest several heuristics and a methodology suitable for the exploratory analysis of large scale real-world data sets, whose underlying patterns are difficult to recover for digitization reasons. Extensive experiments show the benefits of the approach, evaluated with a dual objective: the accuracy of density estimation in the case of outliers or heavy-tailed distributions, and the effectiveness of the approach for exploratory data analysis.

Keywords: Density estimation, Histograms, Model selection, Minimum description length, Exploratory analysis

1. Introduction

Histograms are among the most popular methods used in exploratory analysis to summarize univariate distributions. *Regular histograms* are the simplest savor of histograms to represent a distribution: all bins are of the same width and the only parameter to select is the number of bins. While they are suited to roughly uniform distributions [1], they fail to capture the density of more complex distributions. *Irregular histograms* are non-parametric piecewise constant density estimators that require very few parameters: the number of bins with their widths and frequencies. Several irregular histogram methods have been proposed in the literature, but they often require user-defined parameters, such as the number of bins or the accuracy ϵ at which the data is to be approximated. For example, the minimum description length (MDL) histogram methods [1, 2] automatically choose the number of bins and their widths, but these widths need to be a multiples of a user parameter ϵ. In the context of exploratory analysis, the choice of this parameter is not an easy task, and fully automatic histogram methods are preferable. Several automatic irregular histogram methods have been proposed in the literature, such as the *taut string* methods based on penalized likelihood [3, 4], the *Bayesian blocks* histograms based Bayesian regularization [5] or the *G-Enum* method [6] based on the MDL approach. In a comparison between several regular and irregular histograms methods, the G-Enum method achieves state-of-the-art accuracy for estimated density while being much more scalable than its closest competitors [6]. It is also among the most parsimonious methods, with far fewer intervals than the most accurate alternative methods,

which is an essential feature for exploratory analysis when interpretability is an issue. These properties being in line with our main objective in this paper, we focus on this method.

The G-Enum method [6] extends the MDL method [2] with an automatic choice of ϵ, a fast to compute closed-form evaluation criterion and scalable efficient optimization heuristics. Its modeling space is described on the basis of ϵ-length elementary bins, where each histogram bin consists of a subset of adjacent ϵ-length bins. A granularity parameter is exploited to automatically select the ϵ parameter. Together with efficient linearithmic optimization heuristic, this granulated MDL criterion provides a resilient, efficient and fully automated approach to histogram density estimation. Nevertheless, this method reaches its limits in the case of outliers or heavy-tailed distributions and in the case of real world data sets.

This paper presents an extension of the G-Enum method that exploits the floating-point representation of real number on computers, and a methodology for univariate exploratory analysis of large scale real-world data sets. The first key contributions are related to

- histograms for density estimation:
  - introduction of a space of floating-point bins, as an alternative to equal-width bins,
  - extension of the G-Enum method with new criterion and algorithm,
  - extensive experiments in the case of outliers or heavy-tailed distributions.

However, while the method works very well on challenging artificial data sets with known distributions, preliminary experiments show the limitations of the approach when applied to real-world data sets, where the aim is to provide a better understanding of the data through exploratory analysis. The second key contributions are related to

- histograms for exploratory analysis:
  - characterization of some issues that come with real-world data,
  - proposal of a methodology for effective exploratory analysis using histograms,
  - illustration with several use cases related to challenging real-world data sets.

The rest of the paper is organized as follows. We briefly recall the G-Enum method in Section 2. We illustrate the limit of histogram methods in the case of outliers and discuss possible solutions to push these limits in Section 3. We introduce the notion of floating-point bins in Section 4, and exploit them to suggest an approach named G-Enum-fp in Section 5. We perform extensive experiments with artificial data sets in Section 6. We then propose a methodology for the exploratory analysis of real-world data sets in Section 7, which we evaluate in Section 8. Finally, we give a summary and suggest future work in Section 9.

2. G-Enum method: summary

This section is a brief reminder of the G-Enum method [6].

2.1. Problem formulation

We consider a sample of n observations $x^n = (x_1, ..., x_n)$ on the interval $[x_{min}, x_{max}]$. Let ϵ be the approximation accuracy, so that each $x_j \in x^n$ can be approximated by $\widetilde{x}_j \in \mathcal{X} = \{x_{min} + t\epsilon; t = 0, ..., E\}$ where $E = L/\epsilon$ and $L = x_{max} - x_{min}$ is the '*domain length*' of the data. We expect to have $E \in \mathbb{N}$.

Let $\mathcal{C}$ be the set of possible endpoints for sub-intervals as

$$\mathcal{C} = \{c_t = x_{min} - \frac{\epsilon}{2} + t\epsilon; t = 0, \ldots, E\}.$$

These endpoints define E *elementary bins* of length ϵ, which are called ϵ-bins. They are the building blocks of histogram intervals: each combination of ϵ-bins into K intervals, with K ranging from 1 to E, defines a histogram model. In this range of possibilities, the goal is to select a set of $K-1$ endpoints $C = (c_1, ..., c_{K-1}),\ c_k \in \mathcal{C}$ such that $[c_0, c_K] = [x_{min} - \epsilon/2, x_{max} + \epsilon/2]$ is partitioned into K intervals $\{[c_0, c_1],]c_1, c_2], ...,]c_{K-1}, c_K]\}$ that are well-suited to the actual data distribution. Each interval k has a data count of h_k entries and a length $L_k = c_k - c_{k-1}$, which is a multiple of ϵ:

$$\forall k,\ \exists\ E_k \in \mathbb{N} \text{ such that } L_k = E_k \cdot \epsilon$$

A histogram model is entirely defined by the choice of the number of intervals, the set of endpoints that define them and their data counts. We thus note a histogram model $\mathcal{M} = (K, C, \{h_k\}_{1 \leqslant k \leqslant K})$. The relevance of each model can be measured through different types of MDL criteria, for example using an enumerative criterion.

2.2. Granularity and choice of ϵ

The role of approximation accuracy ϵ has been studied in [6] for the Enum method, both theoretically and empirically. When ϵ tends towards 0, or equivalently E tends towards $+\infty$, the MDL criterion is dominated by the prior terms and the number of intervals decreases asymptotically down to a single interval.

To get rid of this user parameter ϵ, a new model parameter is introduced, that will automatically be inferred. Let G be the granularity parameter. For a given E, the numerical domain is split into G bins ($1 \leqslant G \leqslant E$) of equal width. In practice, the constant $E = 10^9$ is used, which is both close to the limits of the representation of machine integers and allows to obtain very accurate histograms, with an accuracy of up to one billionth of the value domain. Each of these new elementary bins, that are called *g-bins*, is composed of $g = E/G$ ϵ-bins. Each of the intervals of any histogram constructed has then a length that is a multiple of these g-bins. In other words, each interval is no longer composed of a multiple of ϵ-bins but rather composed of G_k g-bins.

This new criterion, which is called **G-Enum** is still very similar to the MDL-based enumerative criterion **Enum** for histograms, as shown in table 1. The resulting method is parameter-free, as it does not depend on any user parameter [1].

2.3. Enum and G-Enum criteria for histogram models

Table 1 recalls the Enum criterion for histogram models and its granulated extension G-Enum. The $\log^* K$ and $\log^* G$ prior terms encode the choice of the number of intervals and of the granularity parameter. They exploit Rissanen's universal prior for integers [7], that favors small integers, i.e. simpler histograms. The $\log \binom{G+K-1}{K-1}$ term encodes the boundaries of the intervals at the granularity precision. The multinomial terms are used to encode the multinomial distribution of the n instances on

[1] *user parameters* (e.g. ϵ) have to be adjusted by the data analyst, *model parameters* belong to the modeling space and are inferred automatically by optimizing a criterion, *technical parameters* are internal constants, used for example as upper-bounds of model parameters (e.g. $E = 10^9$, with $1 \leqslant G \leqslant E$)

Table 1
Term comparison of the Enum and G-Enum criteria

Criterion	Indexing terms	Multinomial terms	Bin index terms
Enum	$\log^* K + \log \binom{E+K-1}{K-1}$	$\log \binom{n+K-1}{K-1} + \log \frac{n!}{h_1!...h_K!}$	$\sum_{k=1}^{K} h_k \log E_k$
G-Enum	$\log^* K + \log^* G + \log \binom{G+K-1}{K-1}$	$\log \binom{n+K-1}{K-1} + \log \frac{n!}{h_1!...h_K!}$	$\sum_{k=1}^{K} h_k \log G_k + n \log \frac{E}{G}$

the K intervals. They rely on an enumerative criterion with appealing optimality properties [8]. The $\sum_{k=1}^{K} h_k \log G_k + n \log \frac{E}{G}$ term encodes the position of the h_k instances of each interval on the $E_k = G_k \frac{E}{G}$ elementary ϵ-bins of the interval.

2.4. Optimization algorithms

For additive criteria such as Enum, a dynamic programming algorithm can be applied to obtain the optimal solution. However, its computational complexity is cubic w.r.t. the number endpoints E, making it impractical in the case of large data sets. To achieve a practicable computational complexity, the Enum method exploits a greedy bottom-up optimization heuristic that starts with the most refined histogram based on ϵ-bins, then merges adjacent intervals until the criterion can no longer be improved. The quality of the model is then improved using post-optimization heuristics, which mainly consist of adding, removing, or moving endpoints around the local optimal solution. Moreover, in the case of the Enum criterion, the optimal endpoints are necessarily close to data points, as demonstrated in [6], which reduces the number of candidate endpoints from E to $O(n)$, resulting in an overall computational complexity is $O(n \log n)$ instead of $O(E^3)$. Experiments show that the accuracy of histograms optimized using these heuristics is indistinguishable from those using the optimal algorithm, while being much faster to compute. As for the G-Enum method, only the powers of two granularities are considered and the Enum algorithm is called $O(\log E)$ times, the computational complexity remaining $O(n \log n)$ since E is a constant.

2.5. Experimental results

We summarize below the results of the comparative experiments performed to evaluate the G-Enum method [6]. The comparison includes the following irregular and regular histogram methods:

- `G-Enum`, the method summarized in this section,
- `Enum`, the base method, with user parameter $\epsilon = 0.01$,
- `NML` histograms [2], with user parameter $\epsilon = 0.01$,
- `Taut string` histograms [3, 9],
- `RMG` histograms [4],
- `Bayesian blocks` [5],
- `Sturges` rule histograms,
- `Freedman-Diaconis` rule histograms [10].

It is worth noting that the Enum and NML methods are similar in that they both require a ϵ user parameter, share the same modeling space and exploit a MDL approach. They differ in terms of optimized MDL code and computational complexity: *Normalized Maximum Likelihood* code and $O(E^3)$ for NML, versus enumerative code and $O(n \log n)$ for Enum. All the other methods are parameter-free.

The histogram methods are evaluated on artificial data sets with known distributions: Normal, Cauchy, Uniform, Triangle, Triangle mixture and Gaussian mixture. The methods are compared on three criteria: accuracy evaluated with the Hellinger distance, parsimony using the number of intervals and computation time. The analysis of the experimental results shows that the G-Enum method achieves state of the art accuracy while being much more parsimonious and faster than its closest competitors.

> "Although rarely the best for each distribution type, `G-Enum` histograms are consistently among the best estimators, and this without the high variability of the other methods. Focusing on irregular histograms, `G-Enum` is certainly among the most parsimonious in number of intervals. For exploratory analysis, this is an important quality because it makes the interpretation of the results easier and more reliable. `G-Enum` is also by far the fastest of irregular methods, making it suitable to large data sets." ([6])

In particular, in the case of the heavy-tailed Cauchy distribution with the largest evaluated data set size ($n = 10^5$) and widest value domain, G-Enum histograms are the most accurate density estimators while being between 10 and 1000 times faster than their accurate competitors.

3. Limits of histogram methods w.r.t. outliers

We first give an illustrative example of the limits of the G-Enum method in the case of outliers, and then discuss possible solutions to push these limits.

3.1. Illustative exemple

Let us consider a data set containing $n = 10,000$ data entries distributed according to a Gaussian distribution $G(\mu = 0, \sigma = 1)$. The range of the numerical domain is $L = (x_{max} - x_{min})$. As $\sigma = 1$, we have $L \leqslant 10$ with high probability. The range of the numerical domain at ϵ accuracy is $E = L/\epsilon$. Let us recall that we have chosen $E = 10^9$ to be compliant with the computer representation of integers using four bytes. As a matter of fact, computer integers are in the value domain $]-\text{INT\_MAX}; \text{INT\_MAX}[$, with $\text{INT\_MAX} = 2^{31} \approx 2.10^9$. Using the $E = 10^9$ precision parameter, the bounds of the histogram intervals are very precise, and the underlying distribution can be very well approximated as the number of data entries n increases.

Let us now assume that we have an outlier data entry in our data set, with value $x_{out} = 10^{12}$. The range of the value domain becomes $L \approx 10^{12}$ and using the same precision parameter $E = 10^9$ amounts to setting $\epsilon \approx 1000$. With this ϵ parameter, the optimal histogram reduces to a histogram with two intervals, consisting of a first interval of width $E_1 = 1$ that contains all the n initial Gaussian data entries in a bin of width 1000, and a second interval of width $E_2 = E - 1$ containing the outlier data entry. The quality of the histogram becomes very poor as the whole data set except one outlier is summarized using one single interval.

Let us note that, to the best of our knowledge, this problem is likely to occur with most alternative histogram methods. In the following we investigate on solutions to push these limits.

3.2. Possible solutions to push these limits

We suggest possible solutions to push the limits of the G-Enum method and summarize their potential benefits and drawbacks.

3.2.1. Use of long integers

The use of computer integers beyond 10^9 could be considered for the number E of ϵ-bins. For example, arbitrary large integers such as python integers could be used. However, this solution is unlikely to work reasonably well for the following reasons.

- this greatly increases computation time, as large integers are not processed at processor level.
- this poses numerical problems in calculating the optimization criterion for integers beyond 10^{15}, since mathematical functions such as logarithm are limited to a mantissa precision of 15 digits,
- this cannot work well when small g-bins are needed to recover the main patterns of a data set, at the expense of very large granularity G to keep the outliers in the numerical domain; indeed, when G tends towards $+\infty$, the number of intervals decreases asymptotically down to a single interval (see Section 2.2).

3.2.2. Removing outliers

Removing outliers before calculating the histogram could be considered. Outlier detection has been widely studied [11] and many methods have been proposed in the literature. There is no generic or universally applicable outlier detection method, and most existing methods require user thresholds, which are difficult to adjust. In the case of exploratory data analysis with no prior knowledge of the data, outlier removal prior to histogram calculation is questionable. For example, in the case of a heavy-tailed distribution with no mean or variance, such as a Cauchy or Lévy distribution, many extreme values could be removed regardless of the user threshold, and the remaining extreme values could still be considered outliers.

The primary purpose of histograms is to provide an initial overview of the data for exploratory analysis without any prior knowledge, and ideally no data should be excluded. The resulting histograms can then be used as building blocks for anomaly and outlier detection methods [12, 13].

3.2.3. Extension to hierarchical histogram models

One solution to cope with outliers consists in extending the G-Enum method to a hierarchical model. A histogram consists in a set of adjacent intervals, whereas a hierarchical histogram consists in a tree of intervals, where:

- each leaf node is an interval,
- each intermediate node can be seen both as an interval, union of its children intervals, and as a histogram, set of its children intervals,
- the root node represents the whole value domain.

Such a hierarchical histogram could potentially cope with outliers. For example, using the data set described in Section 3.1, we could have one root node with three children nodes; the first one for all the Gaussian data entries, the second one with an empty interval and the last one with the outlier. Then the first node could be divided again to produce a standard histogram focused on the Gaussian data entries, without any outlier issue.

This possible solution looks appealing, but its implementation may encounter several problems:

- devising an effective prior for hierarchical models is not an easy task,

- optimizing hierarchical models is known to be difficult, with little hope of achieving optimality efficiently,
- the optimization algorithm may face numerical problems, since many models to be compared may have almost the same cost.

3.2.4. Bi-level heuristic for histograms

A heuristic variant of hierarchical histogram models have been investigated in [14]. The resulting bi-level heuristic exploits a logarithmic transformation of the data to split the data set into a list of data subsets with a controlled range of values. The second level builds a sub-histogram for each data subset and aggregates them to obtain a complete histogram. Extensive experiments have demonstrated the applicability of the method to a wide range of data sets, including the case of outliers or heavy-tailed distributions. However, this method is hampered by some heuristic trade-offs:

- it relies on several hard to tune heuristic thresholds, mainly to split or not the initial data set into a list of data subsets and to deal with tiny value ranges that reach the limit of the precision of the mantissa of real values,
- it relies on a sub-optimal heuristic to split the initial data set into a list of data subsets,
- it requires hard to tune heuristic methods to aggregate the independent sub-histograms, with potentially different granularity parameters, obtained per data subset,
- the overall optimization heuristic is tricky to implement, with a significant computation time overhead,
- an overall evaluation criterion is missing for the bi-level method, which prevents from providing a quality indicator and from post-optimizing the overall histogram or simplifying it wisely.

4. Floating-point bins for histograms

Histogram methods are univariate non-parametric density estimators which provide a summary of the underlying distribution using piece-wide constant densities per interval. These methods are devised with the assumption of data values belonging to $\mathbb{R}$. In practice, the only values that can be observed have to be represented on computers and rely on floating-point representation, with radically different properties compared to values of $\mathbb{R}$. We suggest to exploit this floating-point representation, with the objective of building histograms for data distributions that can be represented on a computer rather that arbitrary distributions on $\mathbb{R}$. In this section, we first recall the format of floating-point representation and analyze some of its properties. We then introduce the definition of *floating-point bins*, an alternative to equal-width bins as building blocks for histogram intervals.

4.1. Floating-point representation

Let us first summarize how real values are encoded on computers using the floating-point representation [15]. Computer real values with double-precision floating-point format are stored on 8 bytes and thus encoded using 64 bits:

- sign: $r_s = 1$ bit,
- exponent: $r_e = 11$ bits,
- mantissa: $r_m = 52$ bits.

The sign bit encodes the sign of the number, -1 or +1. The exponent bits encode exponents between DBL_MIN $= 2^{-1022}$ and DBL_MAX $= 2^{+1023}$, that is between around 10^{-308} and 10^{308}. Exponents 2^{-1023} and 2^{-1024} are reserved for special values. The mantissa bits $\{b_i\}_{1 \leqslant i \leqslant r_m}$ exploit an additional implicit bit of value 1 for the integer part and encode numbers of the form $1 + \sum_{i=1}^{r_m} b_i 2^{-i}$, between 1 and $2 - 2^{-r_m}$. The zero value, which is a singular value in the floating-point representation, is encoded differently as a special value.

Whereas mathematical real values that belong to $\mathbb{R}$ are continuous and unbounded, computer floating-point values are discrete in essence and bounded. They belong to a finite set $\mathbb{R}^{(cr)}$ (where (cr) stands for *computer representation*). The set $\mathbb{R}^{(cr)}$ contains $2^{64} \approx 1.8\ 10^{19}$ distinct values that belong to the finite numerical domain $[-10^{308}, -10^{-308}] \cup \{0\} \cup [10^{-308}, 10^{308}]$, with half the values within $[-1, 1]$. Notice that all computer real values have an approximately constant relative precision w.r.t the exponent, but an absolute precision that exponentially increases around the value 0. More precisely, each power of two range $[2^i, 2^{i+1}[$ contains 2^{r_m} distinct equidistant values, and the distance between successive values doubles each time the exponent is incremented. This translates into a piecewise constant density within each power of two range, and an approximately constant relative density w.r.t the exponent. There are more than 600 orders of magnitude of difference of absolute precision between the largest and the smallest computer real values. To summarize, mathematical real values have translation-invariant density properties all over $\mathbb{R}$. Conversely, the density of floating-point representation values in $\mathbb{R}^{(cr)}$ is heavily peaked around the value 0: it increases exponentially for $x \to 0$ until reaching the underflow and decrease exponentially for $x \to \infty$ until reaching the overflow.

Impact on histograms. While histograms are invariant to translation in $\mathbb{R}$, this is not the case in $\mathbb{R}^{(cr)}$. This is a limitation of the data representation, not of the histogram models. To illustrate this non intuitive behavior, let us take D as a data set in $\mathbb{R}^{(cr)}$ and $t_a(D)$ the data set obtained by translating D by the value a. As the mantissa is limited to around 15 digits, we have for example:

- $\forall D \in [0, 1], t_{10^{15}}(D) = \{a\}$,
- $\forall D \in [10^{15}, 10^{16}], t_1(D) = D$.

And all intermediate behaviors can be observed between these two extremes.

4.2. Floating-point bins

The direct exploitation of floating-point representation to design a histogram modeling space is of great interest, as all the data that can be observed and processed are stored on computers.

Histograms where the length of intervals are multiple of ϵ-bins rely on a constant absolute precision and they cannot cope well with outliers. The G-Enum method builds intervals on the basis of at most $G_{max} = E$ elementary ϵ-bins, with $E = 2^{30} \approx 10^9$. We suggest to extend the method by changing the definition of elementary bins used to build the intervals, replacing the equal-width bins of length $\epsilon = (x_{max} - x_{min})/E$ by floating-point bins of varying length.

Let us first introduce *floating-point bins*, that can be divided into:

- *main bins*
  - *exponent bins* of length 2^i, $B_{E-,i} =]-2^{i+1}, -2^i]$ or $B_{E+,i} =]2^i, 2^{i+1}]$;
  - *central bins* of length 2^i around 0, $B_{C-,i} =]-2^i, 0]$ or $B_{C+,i} =]0, 2^i]$
- *mantissa bins* of equal-width $2^i \times 2^{-m}, m \geqslant 0$ within each main bins of length 2^i.

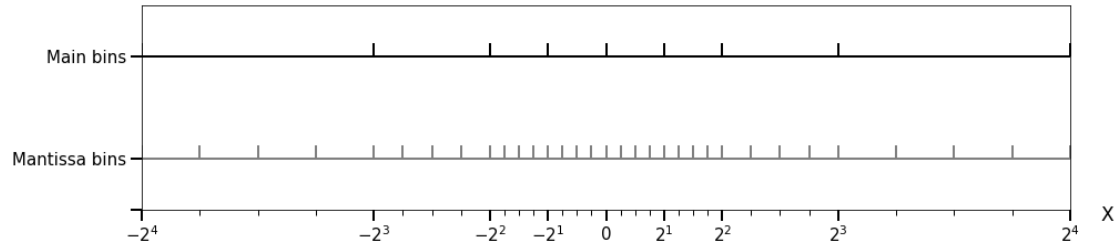


Figure 1. Examples of floating-point bins.

Figure 1 shows an example of main bins and mantissa bins that form a partition of the $]-16,16]$ numerical domain. There are 8 main bins, including two central bins of length $2^1 = 2$, and 32 mantissa bins corresponding to $2^2 = 4$ equal-width bins within each main bin

Standard histograms rely on a set of equal-width elementary bins, which seems well suited to the inference of piecewise constant density estimators. Our objective is to exploit the floating-point representation while keeping close to equal-width bins across the entire value domain. We need to take care of the value 0, which is a singular value in the floating-point representation. For parsimony reasons, we try to avoid unnecessary exponent bins around zero, and for smoothness reasons, we look for a set of floating-point bins that are as close possible as possible to equal-width bins.

We suggest to cover the numerical domain $[x_{min}, x_{max}]$ of the data set to analyze with the most precise possible floating-point bins, in the limit of at most $E = 2^{30}$ elementary bins. To do this, we first cover the data set with as few adjacent mains bins as possible. Let $i_\star$ be the exponent of the largest possible central bins that does not contain any non-zero value of the data set. $i_\star$ is also the smallest exponent among the exponent bins that contain at least one value of the data set. We then cover the numerical domain with exponent bins, the exponents of which are derived from x_{min} and x_{max}, plus potentially one or two central bins around the zero value.

We formalize this below, assuming that $x_{min} < x_{max}$.

- if $0 \leqslant x_{min}$,
  use exponent bins: $\{B_{E+,i}\}_{i_\star \leqslant i \leqslant i_{max}}$ with $i_{max} = \lceil \log_2(x_{max}) \rceil$,
  - if $x_{min} = 0$,
    plus two central bins: $B_{C-,i_\star}$ and $B_{C+,i_\star}$,
- if $x_{max} \leqslant 0$:
  use exponent bins: $\{B_{E-,i}\}_{i_{min} \leqslant i \leqslant i_\star}$ with $i_{min} = \lfloor \log_2(-x_{min}) \rfloor$
  - if $x_{max} = 0$,
    plus one central bin: $B_{C-,i_\star}$,
- if $x_{min} < 0 < x_{max}$,
  use exponent bins for negative values: $\{B_{E-,i}\}_{i_{min} \leqslant i \leqslant i_\star}$ with $i_{min} = \lfloor \log_2(-x_{min}) \rfloor$,
  use exponent bins for positive values: $\{B_{E+,i}\}_{i_\star \leqslant i \leqslant i_{max}}$ with $i_{max} = \lceil \log_2(x_{max}) \rceil$,
  plus two central bins: $B_{C-,i_\star}$ and $B_{C+,i_\star}$.

We get a set of n_B mains bins. If $n_B = 1$, all the data set is contained in one single exponent bin. In this case, we look for the smallest mantissa bin that contains all the data set. This mantissa bin can be split

into mantissa bins of increased precision, until either the total number of elementary bins that cover to data set is greater than 2^{30} or reaching the maximum precision of mantissa ($r_m = 52$). In the end, all the data set is covered with a set of elementary floating-point bins of equal width.

If $n_B > 1$, the data set requires several main bins to be covered. As the maximum number of main bins available using the floating-point representation is given by $(2^{rc} - 2) \times 2 + 2 < 2^{12}$ (cf. Section 4.1), each main bin can be split into at least $2^{30}/2^{12} = 2^{18}$ mantissa bins, that is a relative precision of about four millionths. This gives us the set of elementary floating-point bins that will be used in our modeling space at the maximum precision. We get a set of elementary floating-point bins that are of equal width within each main bin.

All the interval boundaries c_k will be chosen among the boundaries of these elementary bins, with $c_{min} = c_0$ equal to the lower bound of the first elementary bin and $c_{max} = c_k$ equal to the upper bound of the last one. Although the elementary bins have lengths spanning over a range of values exponentially large, they are locally close to equal-width bins, as each elementary bin has a length that is either the same, half or twice that of its adjacent bins.

Finally, like in the G-Enum method, we propose to define the granulated bins by building a hierarchy of bins based on these elementary bins. At the maximal depth, we keep all our elementary bins. Then each time we decrease the depth d, we merge adjacent mantissa bins to obtain super-mantissa bins with one bit less in precision. When the precision $m = 0$ of mantissa bins is attained, that is when we reach the level of main bins, we continue agglomerating the adjacent mains bin using a binary tree, until we obtain one single root bin ($d = 0$).

Let us notice that at any depth of the hierarchy, all the granulated bins are exact floating-point bins, except for the two extrema bins that contain x_{min} and x_{max}, which may be truncated to keep the covered values within $]c_{min}, c_{max}]$. Let us finally define G_d as the number of granulated bins obtained at each level of the hierarchy. We have $2^{d-1} < G^d \leqslant 2^d$, since the number of main bins is not necessarily a power of two and the extrema bins are likely to be truncated at some depths of the hierarchy.

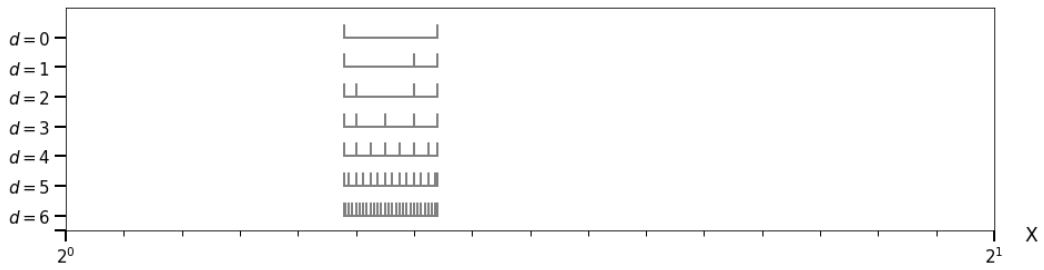


Figure 2. Granulated bins in case of one single main bin.

Example with one single main bin. Let us consider a data set with values in $[1.3, 1.4]$. Only one exponent bin $B_{E+,0} =]2^0, 2^1]$ is enough to cover the data set. Within this exponent bin, the smallest mantissa bin that covers the whole data set is $]1.25, 1.5]$, with a mantissa precision $m = 2$. This allows to choose a set a mantissa bins at precision 32 ($d_{max} + m$=30+2) to cover our data set in a range $]c_{min}, c_{max}]$, with $|c_{min} - 1.3| < 2^{-32}$ and $|c_{max} - 1.4| < 2^{-32}$. For $d = 30$, all the bins have the same length 2^{-32} and the total number of bins is $G_{30} = (c_{max} - c_{min})/2^{-32} < 2^{30}$. As the whole data set is contained in one

single main bin, all the granulated bins have the same length 2^{-2-d} at any depth d, except for the two extrema bins that need to be intersected with $]c_{min}, c_{max}]$. This is illustrated in Figure 2, which shows the partition of the value domain in granulated bins for depths ranging from 0 to 6. For $d = 0$, there one single root bin $]c_{min}, c_{max}]$. For $d = 1$, two mantissa bins of length 2^{-3}, $]1.25, 1.375]$ and $]1.375, 1.5]$ are used, resulting in two granulated bins $]c_{min}, 1.375]$ and $]1.375, c_{max}]$.

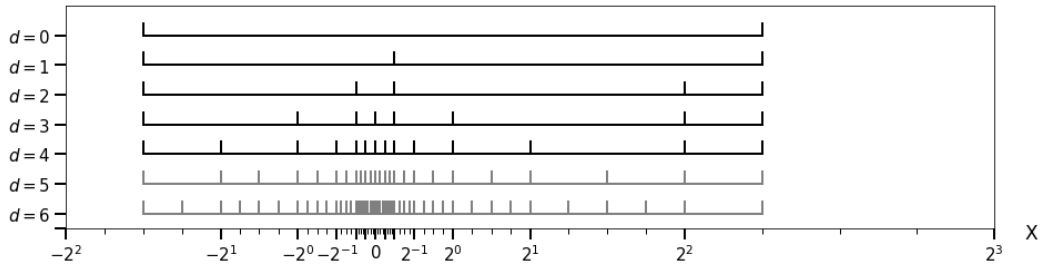


Figure 3. Granulated bins in case of multiple main bins.

Example with multiple main bins. Let us consider a data set with values in $[-3, 5]$, and where the smallest non-zero absolute value is 0.15. We have $x_{min} = -3, x_{max} = 5, i_{min} = 1, i_{max} = 2$ and $i_\star = -3$ (as $2^{-3} < 0.15 \leqslant 2^{-2}$). The value domain is covered using five negative exponent bins $]-2^2, -2^1],]-2^1, -2^0],]-2^0, -2^{-1}],]-2^{-1}, -2^{-2}],]-2^{-2}, -2^{-3}]$, six positive exponent bins $]2^{-3}, 2^{-2}],]2^{-2}, 2^{-1}],]2^{-1}, 2^0],]2^0, 2^1],]2^1, 2^2],]2^2, 2^3]$, plus two central bins $]-2^{-3}, 0],]0, 2^{-3}]$. Altogether, $n_B = 13$ main bins are used, and for $d = 4$, the granulated bins consist of these $G_4 = 13$ bins. For $4 < d \leqslant 0$, the mains bins are grouped by 2, 4, 8, 16, leading to $G_3 = 7, G_2 = 4, G_1 = 2, G_0 = 1$ granulated bins. Conversely, for $d \geqslant 5$, the main bins are split into mantissa bins exploiting $d - 4$ digits for the precision of the mantissa. This is illustrated in Figure 3, which shows the partition of the value domain in granulated bins for depths ranging from 0 to 6. For $d = 2^{30}$, each exponent or central bin is divided into 2^{26} equal-width mantissa bins, ranging from the smallest absolute length 2^{-29} in $]-0.25, 0.25] =]-2^{-2}, -2^{-3}] \cup]-2^{-3}, 0] \cup]0, 2^{-3}] \cup]-2^{-3}, -2^{-2}]$ to the largest absolute length 2^{-24} in $]4, 5]$.

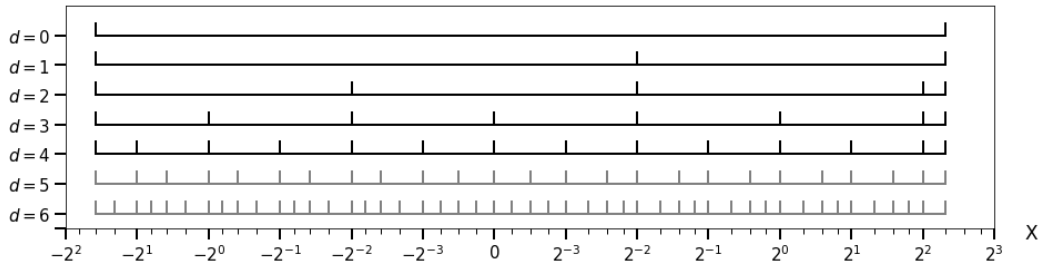


Figure 4. Granulated bins in case of multiple main bins, using a logarithmic scale.

Figure 4 presents an alternative view of Figure 3, keeping the linear scale within the central bins and exploiting a logarithmic scale for the negative and positive exponent bins. This shows that the lengths of

the granulated bins are quite balanced when the relative precision of the interval boundaries is considered rather than their absolute precision. This also suggests an appealing visualization for histograms related to data sets with a dynamic range of values both in the negative and positive domains.

5. G-Enum-fp histogram method

In the previous section, we have exploited the floating-point representation of real values to introduce an alternative definition of the elementary bins used as building blocks for a new histogram method called **G-Enum-fp**. We first summarize the principles of this new method, then summarize its specific components.

5.1. Principle

The G-Enum method exploits a representation space based on elementary equal-width bins, using a granularity parameter G to explore simplified versions of this representation space. It exploits these elementary bins as building blocks that provide a set of predefined bounds, from which the bounds of the histogram intervals are chosen.

The main novelty of the G-Enum-fp method is to replace the elementary equal-width bins of G-Enum with the floating-point bins and their granulated hierarchy introduced in Section 4. This makes a radical difference regarding the impact of the granularity parameter $G, 1 \leqslant G \leqslant E$:

- with the G-Enum method, $E = 10^9$ is considered huge, but it sets a limit of one billionth of the value domain for the smallest interval, which is harmful in event of outliers (see Section 3.1),
- with the G-Enum-fp method and the same E, this limit is extended by more than six hundred orders of magnitude (see Section B.2).

In addition to this major change, the G-Enum-fp method extends the modeling space and the optimization algorithms of the G-Enum method to take full account of the particularities of floating-point bins. To begin with, the bounds of the entire numerical domain are explicitly specified as hyper-parameters. The other difference related to the floating-point representation is the management of the singularity around 0, which relies on the choice of a central bin, treated as an additional model parameter. These extensions are summarized in the following subsections and detailed in Appendix A. Some properties of the G-Enum-fp method are also discussed in Appendix B.

5.2. Specification of domain bounds

With the G-Enum method, the domain bounds are implicitly derived from the data using $[x_{min} - \epsilon/2, x_{max} + \epsilon/2]$. With the G-Enum-fp method, the domain lower and upper bounds are explicitly defined using hyper-parameters that belong to the modeling space:

- main bin containing the domain lower bound,
- main bin containing the domain upper bound,
- central bin exponent of the domain if necessary,
- digit precision used for mantissa bins,
- mantissa bins containing the domain bounds.

These domain lower and upper bounds are inferred once for all, before optimizing the histogram. An evaluation criterion is obtained using a MDL-based approach. Its optimization consist of two steps:

- recover the extreme values x_{min} and x_{max} using a loop over the data set in $O(n)$,
- encode a lower bound of x_{min} and an upper bound of x_{max} using the floating-point representation and optimize the number of digits used for the mantissa, in $O(r_m)$ where $r_m = 52$ is the maximum number of bits in the mantissa.

The details regarding the specification and optimization of these hyper-parameters are given in Appendix A.2.

5.3. Choice of the central bin

With the G-Enum-fp method, we introduce a new parameter i_{cen} to choose the exponent of the central bin used to obtain a representation space based on floating-point bins. In fact, we can choose any value of i_{cen} that conforms with the domain bounds, between $i_{cen_{min}}$ corresponding to the central bin exponent of the domain, and $i_{cen_{max}}$ which allows us to contain the domain lower and upper bounds. With $i_{cen} = i_{cen_{min}}$, we obtain a *maximally floating-point representation*, as the related exponent bins extends over a wide range of values. With $i_{cen} = i_{cen_{max}}$, we obtain a *maximally equal-width representation*, as the bins considered are of equal-width.

Optimizing the exponent of the central bin is mainly a matter of calling the G-Enum optimization algorithm twice, keeping its overall computational complexity:

- optimize a first histogram $M(i_{cen_{min}})$ for $i_{cen} = i_{cen_{min}}$, corresponding to the *maximally floating-point representation*,
- search for the largest value $i_{cen_{opt}}$ of i_{cen} in $[i_{cen_{min}}, i_{cen_{max}}]$ which maintains the same partition of the data set into intervals as in $M(i_{cen_{min}})$, with interval endpoints recoded on the basis of $i_{cen_{opt}}$,
- optimize a second histogram $M(i_{cen_{opt}})$ for $i_{cen} = i_{cen_{opt}}$ and keep this histogram if its evaluation criterion is better than that of the first histogram.

The introduction of this new parameter i_{cen} and its optimization are detailed in Section A.3.

5.4. G-Enum-fp criterion

Table 2
G-Enum-fp criterion

Criterion	Indexing terms	Multinomial terms	Bin index terms
G-Enum-fp	$\log^*(1 + i_{cen_{max}} - i_{cen}) + \log^* K + \log^*(1 + d) + \log \binom{G_d + K - 1}{K - 1}$	$\log \binom{n + K - 1}{K - 1} + \log \frac{n!}{h_1! \ldots h_K!}$	$\sum_{k=1}^{K} h_k \log E_k$

Table 2 shows the G-Enum-fp criterion for the parameters of histogram models. Compared with the G-Enum criterion recalled in Table 1, the only differences concern the indexing terms:

- the term $\log^*(1 + i_{cen_{max}} - i_{cen})$ is used to encode the new parameter i_{cen},
- the exponent d of the granularity ($G = 2^d$) is encoded instead of the granularity G itself,
- the exact number of considered floating-point bins G_d, $2^{d-1} < G_d \leqslant 2^d$ at a given granularity is used instead of $G = 2^d$ in the case of equal-width bins.

In addition, new hyper-parameters have been introduced for the domain bounds (see Table 4 for the corresponding criterion). Note that the G-Enum-fp method is parameter-free, as all its parameters and hyper-parameters belong to the modeling space.

6. Experiments with artificial data sets

In this section, we evaluate the G-Enum-fp method using artificial data sets with known underlying data distribution. We focus on resistance to outliers and heavy-tailed distribution, beyond the limits of state-of-the art methods. A binary standalone implementation of the method is available here: http://marc-boulle.fr/khisto/.

6.1. Evaluation protocol

Metrics. The accuracy of histograms for density estimation is evaluated using the Hellinger distance to the original model density, which is known in the case of artificial data sets. The Hellinger Distance (HD) $H(p,q)$ for p,q being probability density functions, is defined as

$$H(p,q) = \frac{1}{\sqrt{2}}\sqrt{\int (\sqrt{p(x)} - \sqrt{q(x)})^2 dx}$$

A HD close to 0 indicates a strong similarity between probability distributions. The HD measures reported are obtained via numerical integration to estimate the probability distributions of the model density and of the histogram that models it.

The number of intervals per histogram is also collected.

Histogram methods. Histogram methods are limited in the case of outliers or heavy-tailed distribution, as indicated in Section 3. We evaluate the G-Enum-fp method in these challenging cases. We also report the results obtained by the G-Enum method, chosen as a strong baseline, especially in the case of heavy-tailed distributions and large data sets (cf Section 2.5). Since our aim is to push the limits of histograms with data set sizes and value domain widths several orders of magnitude larger than those evaluated in [6], the G-Enum method is in fact the most appropriate method that can be used for comparison purposes.

Data sets. As a sanity check, we first evaluate the behavior of the methods in the case of common distributions, using the uniform, normal and a mixture of normal distributions. The results of these experiments are presented in Appendix C. We then analyze the case of data sets with outliers and heavy-tailed distributions using the Lévy distribution and a pathological mixture of lognormal distributions. The results of these experiments are presented in the following subsections.

6.2. Normal distribution with an outlier

The objective of this experiment is to evaluate the impact of an outlier on the quality of the built histograms. We exploit a data set of size $n = 10,000$ generated from a normal distribution $\mathcal{N}(1, 0.1)$. We add one outlier with value $v_{out} = 2^i, 0 \leqslant i \leqslant 34$ from $v_{out} = 1$ to $v_{out} = 2^{34} \approx 1.7\ 10^{10}$. The experience is repeated 100 times, which represents 3,500 data sets. We collect the Hellinger distance and number

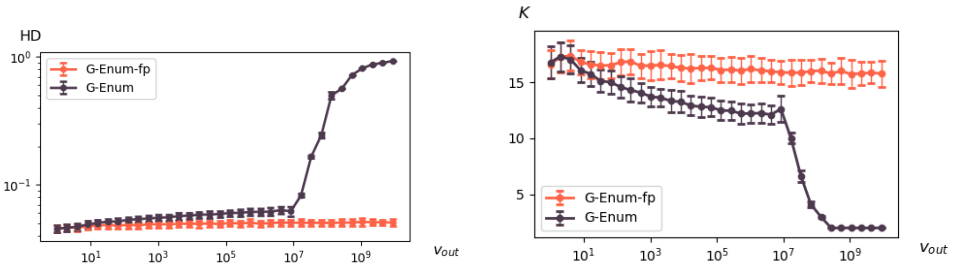


Figure 5. Hellinger distance and number of intervals for $\mathcal{N}(1, 0.1)$ with an outlier.

of intervals. The Hellinger distance is computed using the underlying $\mathcal{N}(1, 0.1)$ distribution, assuming that the impact of one outlier among $n = 10,000$ instance should be negligible.

The overall results are reported in Figure 5. They show that the G-Enum is rather resilient to the outlier on a large scale of values, up to about 10^7, that is more than one million times the range of values of the underlying distribution. The built histograms have a slowly decreasing quality, from around 17 intervals without outlier, down to 12 intervals for $v_{out} \approx 10^7$. For more distant outliers, the 10^9 equal-width elementary bins are no longer sufficient to accurately estimate the underlying distribution. In the end, the histogram consists of two intervals, the first one exploiting one single elementary ϵ-length bin and containing all the normal values, the second one containing only the outlier.

The G-Enum-fp method benefits from its floating-point representation to be highly resilient to the outlier. Beyond $v_{out} > 2$, the G-Enum-fp method always exploits the maximally floating-point representation corresponding to a minimum central bin exponent. As the outlier becomes more distant, the G-Enum-fp method exploits an increasing number of exponent bins, up to around 35 for the largest outlier value $v_{out} = 2^{34}$. The first exponent bins can then be split accurately into mantissa bins, allowing an accurate approximation of the underlying density whatever be the distance of the outlier, which is contained alone in one large interval. We conducted additional experiments with a *googol* outlier ($v_{out} = 10^{100}$). The number of exponent bins necessary to cover the value domain increases up to 334, resulting in largest cost for the prior terms of the G-Enum-fp. The method is still very accurate, using 14 intervals to approximate the normal distribution plus the outlier, instead of 15 for $v_{out} = 10^{10}$.

6.3. *Lévy distribution*

The objective of this experiment is to compare the behavior of the methods in the case of a heavy-tailed distribution. We exploit the Lévy distribution that is *pathological*, having neither mean nor variance. We also evaluate the scalability of the methods, by generating samples of size $n = 10^i, 1 \leqslant i \leqslant 9$. Note that the range of data set values increases very quickly with the sample size, with maximum values greater than 10^{18} for one billion data points. The experiment is repeated 10 times and we collect the Hellinger distance and number of intervals per sample size.

The overall results are reported in Figure 6. The G-Enum method is not able to produce an accurate approximation of the underlying density for n above 10^3. Beyond this threshold, even 10^9 equal-width elementary bins are not sufficient to approximate accurately the Lévy distribution. Indeed, the bins necessary to cover the tails of the distribution become too large for a correct approximation around the median value, which contains most of the probability mass.

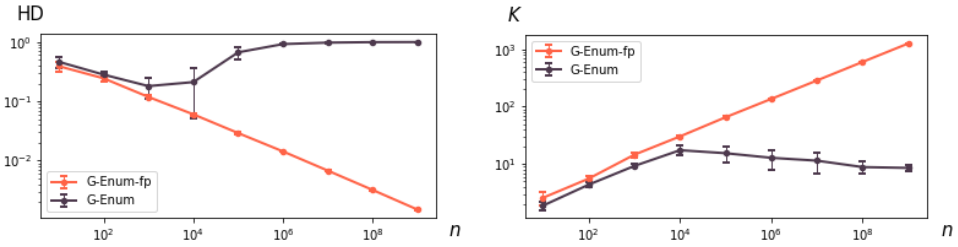
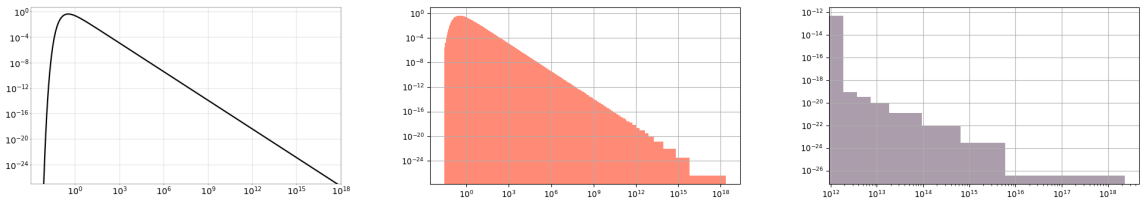


Figure 6. Hellinger distance and number of intervals for a Lévy distribution.

The G-Enum-fp method exploits up to 66 exponent bins to cover the huge range of values for $n = 10^9$, and it keeps an approximately constant relative precision with its mantissa bins, whatever be the position of the interval boundaries. It is able to continuously improve the approximation of the underlying density as n increases, building more and more intervals.



(a) Density. (b) G-Enum-fp histogram. (c) G-Enum histogram.

Figure 7. Density and histograms for a Lévy distribution, with $n = 10^9$.

Figure 7 shows an example of the histograms obtained by each method for $n = 10^9$. The histograms are displayed using a $\log \times \log$ scale for the interval boundaries on the X axis and their densities of the Y axis. The G-Enum-fp histogram that accurately approximates the Lévy distribution consists of 1,253 intervals with lengths ranging from 6.1×10^{-4} to 2.3×10^{18}, frequencies from 8 to 16.03×10^6 and densities 3.4×10^{-27} to 0.46.

6.4. *Lognormal mixture distribution*

The objective of this last experiment is to push the G-Enum-fp to its limits, using a *pathological* mixture of heavy-tailed distributions. We generate data sets of size $n = 10^i, 1 \leqslant i \leqslant 9$ using a mixture of ten lognormal distributions.

$$\sum_{i=1}^{10} \frac{1}{10} \log \mathcal{N}(10^i, \sqrt[10]{10^i}) \tag{1}$$

The experiment is repeated 10 times and Table 9 reports the mean and standard deviation of the number of intervals per sample size. With this pathological distribution, the range of values is enormous, from 10^1 to 10^{24} for data sets with one billion data points, and we were unable to calculate the Hellinger

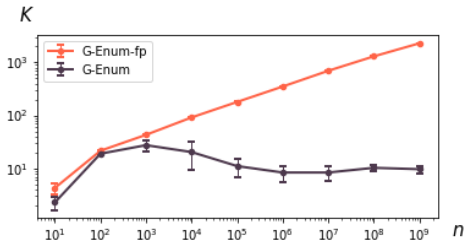
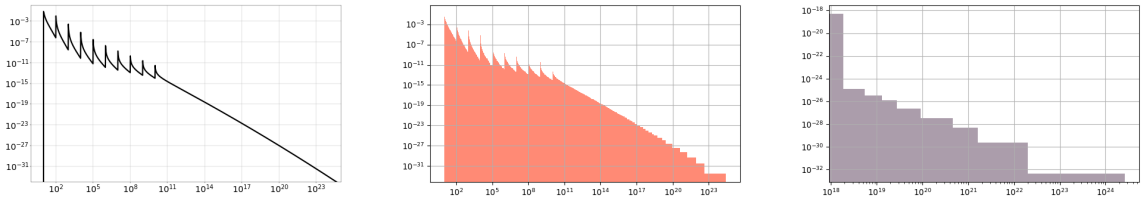


Figure 8. Number of intervals for a mixture of lognormal distributions.

distance due to numerical problems. Due to this huge range of values, the G-Enum method fails to correctly approximate the underlying distribution and ends with histograms containing about ten intervals. On the opposite, as with the Lévy distribution, the G-Enum-fp method continuously improves the approximation of the underlying density as n increases, building more and more intervals.



(a) Density. (b) G-Enum-fp histogram. (c) G-Enum histogram.

Figure 9. Density and histograms for a mixture of lognormal distributions, with $n = 10^9$.

Figure 9 displays an example of the histograms obtained by each method for $n = 10^9$. The G-Enum-fp histogram that accurately approximates the underlying distribution consists of 2,295 intervals with lengths ranging from 4.8×10^{-4} to 2.6×10^{24}, frequencies from 8 to 90.5 million and densities 3.0×10^{-33} to 0.13.

7. Histograms for exploratory analysis

In this section, we apply the G-Enum-fp method to real-world data sets. However, histograms may not be directly useful for the exploratory analysis of real data, as shown in a first example. We then propose heuristics to better process real data and suggest a methodology for using histograms in the context of exploratory analysis. This methodology is illustrated using some standard data sets.

7.1. Applying histograms to real-world data sets

We apply the G-Enum-fp method to the petal length variable of the iris data set [16]. The obtained histogram is presented in Figure 10

The resulting combed histogram consists of 51 intervals, 25 of them containing one single value and of width 3×10^{-8}. In fact, the G-Enum-fp method is a density estimator, and it reveals that the iris data

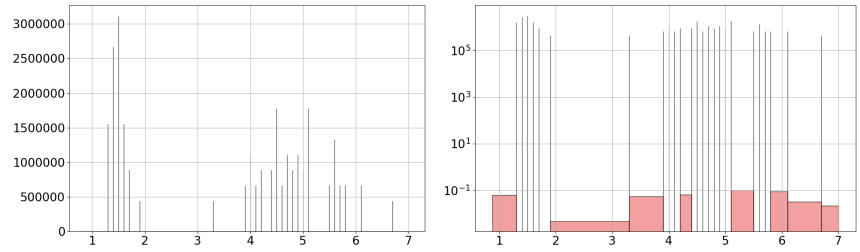


Figure 10. Iris petal length histogram, with the density on a linear scale (left) and a logarithmic scale (right).

set consists mostly of discrete data, leading to a list of density peaks. This is actually a good density estimate, as the petal length variable is recorded with a decimal precision of 0.1, with the 150 instances containing only 43 distinct values.

However, this combed histogram is not very useful for exploratory analysis. Usually, this problem of truncated data is solved by practitioners by setting an adequate minimum bin width, the *truncation gap*, for the histogram intervals. This can be a tricky task for new data for which there is no prior knowledge to guess this truncation gap, and tedious trial and error testing may be required.

7.2. Heuristic to deal with truncated data

We propose an adaptation of the G-Enum-fp method to process integer data, then a heuristic to automatically process truncated data.

7.2.1. Dealing with integer data

First, note that all integers n that can be represented on a computer belong to floating-point bins $]n-1, n]$: either the central bins $]-2^0, 0]$ and $]0, 2^0]$, the exponent bins $]-2^1, -2^0]$ and $]2^0, 2^1]$, or mantissa bins of width at least 2^0 for the exponent bins with larger exponents. To adapt the G-Enum-fp method to integer data, we suggest to exploit the subset of floating-point bins that are compatible with integers, that is all floating-point bins larger than 1. There are few impacts on the G-Enum-fp method:

- the domain bounds represented by the hyper-parameters need to be integers,
- the exponent of the central bin is at least 0,
- the granulated bins are constrained to be of width at least 1.

In the main algorithm (cf. Section A.4), at each depth d of the hierarchy of granulated bins, the main bins of width $2^e, e \geqslant 0$ are split into $\max(2^d, 2^e)$ granulated bin. This results in a lower value of G_d, the number of granulated bins G_d that cover the data set, in the *indexing terms* of the G-Enum-fp criterion. Apart from these changes concerning the granulated bins to consider and the value of G_d, the method is the same.

This extension of G-Enum-fp to integer data allows to process truncated data with a truncation gap of 1, which translates into floating-point bins with a minimum width of 2^0. Note this can be applied in the same way to any floating bins of width at least 2^i, to process any truncated data which truncation gap is a power of two.

7.2.2. Dealing with truncated data

Our objective is to automatically process truncated data to facilitate the task of exploratory analysis. We then suggest a three-steps *truncation management heuristic* (TMH):

(1) detection of truncated data,
(2) calculation of the truncation gap,
(3) construction of a histogram adapted to truncated data.

Detection of truncated data. Let us define a *peak* as a histogram interval which density is greater than that of its previous and next intervals, a *spike* as a peak containing one single value and a *singularity* as a spike which previous and next intervals are empty. For example, the combed histogram in Figure 10 contains 25 spikes, 14 of which being singularities. We assume that spikes are a signature of truncated data and choose to trigger the next steps in the presence of at least one spike. We expect this very simple criterion to correctly identify most of the relevant cases while avoiding unnecessary additional computation time in the other cases.

Calculation of the truncation gap. After sorting the n data entries $x_1, \ldots, x_n$ by increasing values, we collect the $(n-1)$ variations of values $\delta x_i = x_{i+1} - x_i, 1 \leqslant i \leqslant n-1$, and compute a histogram from these variations of values.

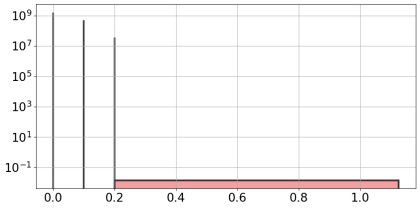


Figure 11. Histogram of variations of values for petal length.

As an example, the resulting variation histogram is shown in Figure 11 in the case the petal length variable of the iris data set. We propose to detect the following *truncation pattern* within the variation histogram to confirm whether the data is truncated:

- the first interval contain only the value 0,
- the second interval is empty,
- the third interval has a strictly smaller length than the second interval.

This pattern is present in Figure 11, which contains two spikes related to the variation of values 0.1 and 0.2. Note than we do not impose that the third interval of the variation histogram be a spike, in order to be resilient to potential rounding errors. We finally exploit the truncation pattern to calculate the truncation gap, as the mean value contained in the third interval, that is the averaged minimum distance between two consecutive distinct values.

Construction of a histogram adapted to truncated data. If a non-zero truncation gap γ_t is available, we exploit the adaptation of the the G-Enum-fp method to integer data described in Section 7.2.1. We first choose a binary truncation gap γ_{bt} as close as possible to the truncation gap, according to $\gamma_{bt} = 2^{i_{bt}}$ with $i_{bt} = \lceil \log_2(\gamma_t) \rceil$. We then transform the initial data so that they conform to the binary truncation gap, then we compute the histogram using the method described in Section 7.2.1 with a minimum floating bin of width γ_{bt}, and finally reverse transform the obtained interval bounds to conform with the initial value domain.

To transform the initial data and get values that are all multiples of the binary truncation gap, we project the data on the upper bounds of the floating-point bins of width γ_{bt} that contain each data entry:

$$y_i = \lceil x_i/\gamma_t - 1/2 \rceil \gamma_{bt}. \tag{2}$$

Symmetrically, the reverse transformation of the interval bounds is calculated using

$$x_i = (y_i/\gamma_{bt} + 1/2)\gamma_t. \tag{3}$$

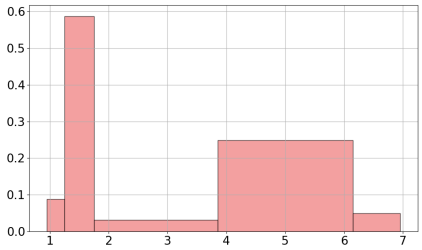


Figure 12. Iris petal length histogram, taking into account truncated data.

For example, the obtained histogram related to the petal length variable of the iris data set is presented in Figure 12. It consists in five intervals, which looks reasonable given that the data set contains only 150 data entries. Its shape seems consistent with the knowledge available concerning the iris data set. It is mixture a three classes of iris flowers, setosa, versicolor and virginica, of small, medium and large sizes, with the petal lengths in $[1.0, 1.9]$ for setosa, in $[3.0, 5.1]$ for versicolor and in $[4.5, 6.9]$ for virginica.

7.3. Methodology for exploratory analysis

Our first objective in this paper was to devise a histogram method that could be applied automatically for the exploratory analysis of any real-world data set. While this goal seems attained in the case of artificial data set where the data generation process is controlled (see Section 6), it may not be achievable with real-world data sets. During the digitization process, the data may exhibit a wide range of tricky patterns beyond rounding or truncation issues, meaning that

> "the digitized structure of the data is a much more robust feature than the statistical structure of the original data. In such cases, an uninvertible transformation has been applied to the data, and information has been irrevocably lost." ([17])

The method based on an optimal histogram algorithm [17] allows to identify digitization problems, which is useful as "it may be desirable for researchers to know that information has been discarded". Beyond this useful feature, our relaxed goal is to help the data analyst filter the digitized structure of the data and discover its statistical structure. We suggest first computing a series a histograms of varying granularities, the finest grain allowing to discover local patterns and the coarsest grain allowing to focus on global patterns. We then provide a list a indicators per histogram to facilitate the task of exploratory analysis.

It is worth noting that this methodology may also be used with other histogram methods. However, its deep integration with the G-Enum-fp method brings several interesting advantages: resistance to outliers and heavy-tailed distributions (cf. Section 6), fine-tuning of the criterion and algorithms in the case of truncated data (cf. Section 7.2.1), and reuse of the intermediate histograms obtained at each depth of granularity as a by-product of the main G-Enum-fp optimization algorithm (cf. Section 7.3.1).

7.3.1. Series of histograms

We optimize a first histogram using the G-Enum-fp method and keep it even in the case of singularities, as it may bring some information regarding digitization problems (as in [17]). We then apply the TMH heuristic to build an adequate histogram if the data are detected as truncated. The current histogram, obtained for an optimal depth of granularity d_{opt}, can be seen at coarser grains for all the intermediate depths d, $0 \leqslant d \leqslant d_{opt}$: we collect all these intermediate histograms evaluated along the optimization trajectory. In order to focus on interpretability and to automatize the exploratory analysis as much as possible, we apply a *singularity removal heuristic* (SRH) by discarding the finest grained histograms having singularities (spikes with two surrounding empty intervals). We could relax the removal criterion by considering spikes or even empty intervals, but this might destroy some potentially useful information.

To summarize, we get a list of *interpretable histograms* by decreasing depth of granularity, $d_{opt_{inter}} \geqslant d \geqslant 0$, plus potentially a *raw histogram* if the first optimized histogram was not interpretable.

7.3.2. Indicators per histogram

To facilitate the exploration of the list of interpretable histograms, we provide the following indicators based on elementary patterns: number of intervals, peaks, spikes and empty intervals. We also introduce a last indicator based on the level-fp criterion (see Section B.4) that is the percentage of information kept in an interpretable histogram compared to the finest grained interpretable histogram:

$$\%\text{information}(M) = \frac{\text{level-fp}(M)}{\text{level-fp}(M_{d_{opt_{inter}}})}. \tag{4}$$

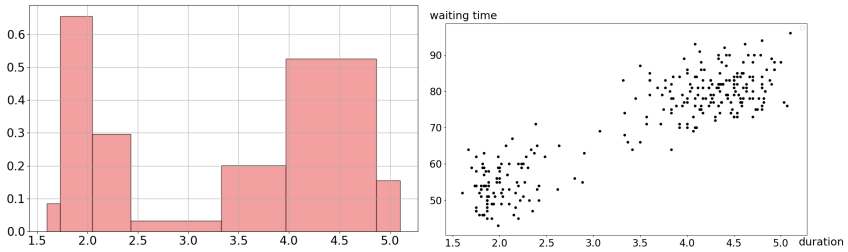
7.4. Illustration

We illustrate the exploratory analysis methodology introduced in this section using some simple data sets. Extensive evaluation is performed in Section 8.

7.4.1. Old Faithful geyser

This data set contains 272 data entries related to eruptions of the Old Faithful geyser [18], with duration and waiting time between eruptions. The duration is used in [9] to illustrate the difficulties of correctly identifying density peaks applying automatic histogram methods on heavily rounded real data. Whereas histograms with two peaks are usually expected, nine automatic histogram methods evaluated in [9] largely disagree on the shape of the histogram, building 5, 11, 19, 23, 37, 42, 111, 143 or 149 intervals.

Similarly, the G-Enum-fp method fails to identify the underlying statistical pattern and reveals the digitization structure with a histogram having 71 intervals, including 35 spikes. The TMH method identifies a truncation gap of 0.001 and outputs the interpretable histogram shown in Figure 13a. Its contains two density peaks as expected, and its shape seems to correspond with the underlying statistical density, as suggested by a visual inspection of the scatterplot in Figure 13b.



(a) Histograms of eruption durations. (b) Duration vs waiting time.

Figure 13. Old faithful geyser.

Note that the truncation pattern seems rather surprising for this data set, as only two pairs of successive values are separated by 0.001, while 31 are separated by 0.016 and 57 by 0.017. As $1/60 \approx= 0.0166$ and after inspecting the data in their plain text format, we assume that the data were recorded with a precision of one second, then transformed to decimal minutes according to $x = m + s/60$ and finally truncated using 3 decimal digits. The digitization process thus includes three steps: recording of observations with limited precision, transformation then truncation of the data. This leads to fine-grained digitization patterns that dominate the underlying statistical patterns. Using the methodology suggested in Section 7.3 provides useful information and allows to recover the statistical structure of the data beyond their digitization structure.

7.4.2. Adult

The adult data set [16] contains 48,842 data entries extracted from a census database. We apply our method to the age variable, which is an integer variable as ages are commonly recorded with one year precision. The G-Enum-fp method builds a raw histogram with 143 intervals, 70 of which being spikes, including 67 singularities. The TMH method correctly identifies a truncation gap of 1 and outputs the interpretable histogram shown in Figure 14.

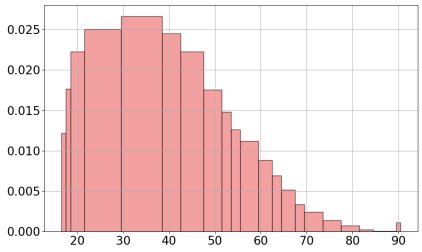


Figure 14. Age in adult data set.

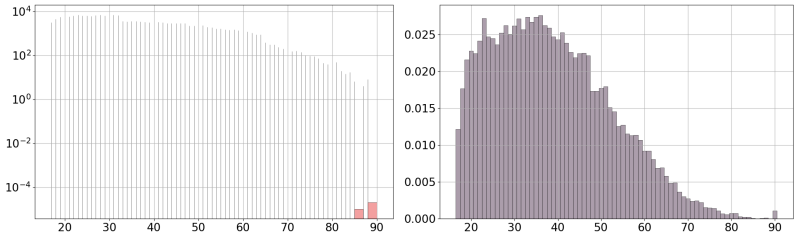
Note that there is a surprising increase of density at age 90. This pattern is confirmed by a close examination of the three last intervals of the histogram, with 39 people between 82 and 84 years of age, 17 between 85 and 89 years of age, and 55 for 90 years of age, the maximum age in the data set. This may indicate that people over the age of 90 were all recorded with the age of 90.

Discussion. For illustration purposes, the raw histogram obtained before applying the TMH heuristic is shown in Figure 15a. This raw histogram is a very accurate density estimator for this data set containing

48,842 data entries for only 74 distinct values. All ages except two are distributed in bins of very small width separated by empty bins almost a year wide. The two exceptions contains too few data entries to be isolated: one data entry for age 86 and two for age 89. Although this histogram is very accurate, it is of little interest for exploratory data analysis.

A regular histogram with a suitably chosen equal-width (value 1 year) is also presented in Figure 15b. It gives the overall shape of the data, but is very noisy, compared to the interpretable histogram obtained using the TMH heuristic (see Figure 14) which recovers a smooth version of the distribution.

Let us note that histograms for mixed discrete-continuous data have been proposed [19], with a user-defined threshold (e.g. 5) for detecting discrete values and a histogram method for the rest of the data. In the case of the adult data set, this approach would result in a histogram similar to that shown in Figure 15a, as most values would be detected as discrete. It is not suitable in the case of truncated data, where the discrete nature of the data is an artifact of the digitization process, masking the underlying statistical patterns of interest.



(a) Raw histogram (b) Regular histogram

Figure 15. Raw histogram (log scale) and regular histogram (equal-width=1) of age in adult data set.

7.4.3. Forest cover type

The forest cover type data set [16] contains 581,012 observations (30×30 meter cells) from a geological survey. We analyze the variable *horizontal distance to nearest roadway*. The raw histogram contains 10,992 intervals, revealing serious digitization issues. The TMH method correctly identifies a truncation gap of 1 related to a one meter precision, but it outputs a histogram that is still not interpretable, with 251 singularities. Once the SRH algorithm is applied to get the 1^{st} interpretable histogram, the result has a globally understandable, albeit comb-like shape, as shown in the Figure 17a. In fact, these data suffer from at least two effects of digitization. The data are recorded with a precision of one meter, which is rather accurate but results in the accumulation of data entries around integer values. And the nature of the observations, based on a mesh of square cells of 30×30 meters, is likely to introduce local correlations.

Applying the methodology introduced in Section 7.3, a series of histograms is collected for all the coarse-grained depths of granularities, starting from the 1^{st} interpretable. The indicators collected per histogram are displayed in Figure 16. They suggest to exploit a two times coarsened histogram to eliminate the comb-like pattern while preserving most of the information. The resulting unimodal histogram shown in Figure 17b is rather smooth and suitable for an easy interpretation, while keeping 98.9% of the information.

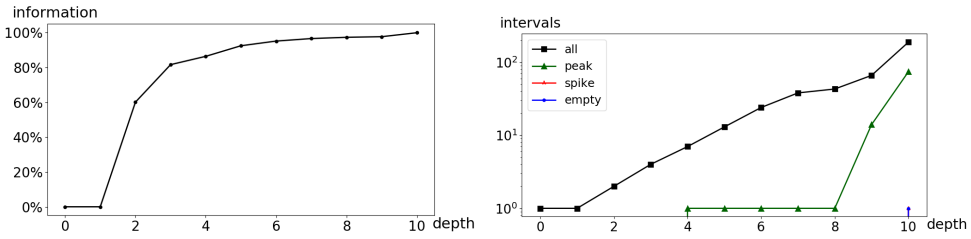
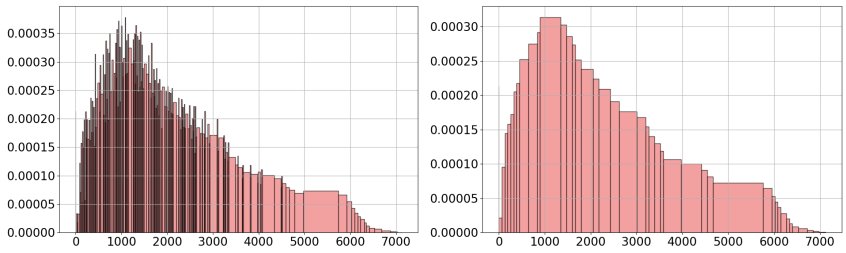


Figure 16. Forest cover type: indicators per depth of granularity.



(a) 1st interpretable histogram. (b) 3rd interpretable histogram.

Figure 17. Horizontal distance to nearest roadway in forest cover type data set.

8. Evaluation with large scale real-world data sets

In this section, we evaluate the floating-point histograms as well as the methodology introduced in this paper for exploratory data analysis, using several large scale real-world data sets.

8.1. Evaluation protocol

Histograms are widely used in exploratory data analysis (EDA) [20] as visualization tools in the data discovery process. However, in the case of challenging real-world data sets, they are difficult to use in practice. Even state-of-the-art histograms are hardly usable in the following cases:

- they reach their limit in the case of outliers or heavy-tailed distributions (cf. Section 3),
- they produce accurate but useless comb-shaped histograms in the case of truncated data (cf. Section 7.4.2),
- they cannot scale with very large data sets or very large value domains (cf. Section 2.5).

The purpose of this section is to assess whether the proposed exploratory methodology, based on the G-Enum-fp method, is capable of pushing the limits of the effective use of histograms for EDA. The approach is evaluated using several real world data set that combine multiple challenges: heavy-tailed distribution, integer data, large scale, complex patterns. As there is no ground truth in EDA, the evaluation cannot be measured objectively. Instead, the patterns recovered using the proposed approach are compared to the prior knowledge available for each data set.

8.2. Moon crater Salamuniccar database

The moon crater Salamuniccar database [21] [2] is a catalog of 78,287 lunar crater impacts. We exploit this database to analyze the distribution of the radius of the craters. The raw histogram output by the G-Enum-fp method contains 2,497 intervals, including 1,204 spikes and 325 singularities. The TMH method identifies a truncation gap of 3×10^{-5}, which looks dubious, and the SRH method discards the 140 remaining singularities to obtain the 1st "interpretable" histogram. This highly combed-shape histogram still contains 249 intervals, including 97 peaks, which is hardly useful for exploratory analysis. These numerous peaks could be explained by the data collection process described in [21], which states that the catalog is globally complete only for crater diameters greater than 8 km, and that the data were recorded using several instruments, some with an accuracy of about one meter and others with a resolution of about 100 meters/pixel. These characteristics are consistent with the data in their plain text format, which consist of less than 3,200 distinct values recorded to a decimal precision of 6 digits.

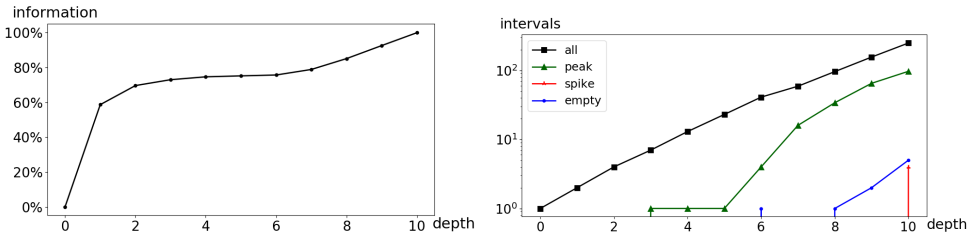
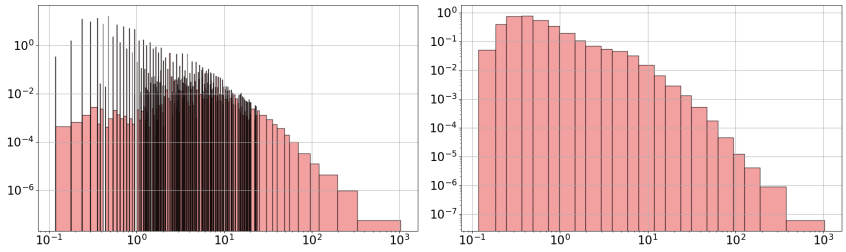


Figure 18. Moon crater Salamuniccar: indicators per depth of granularity.

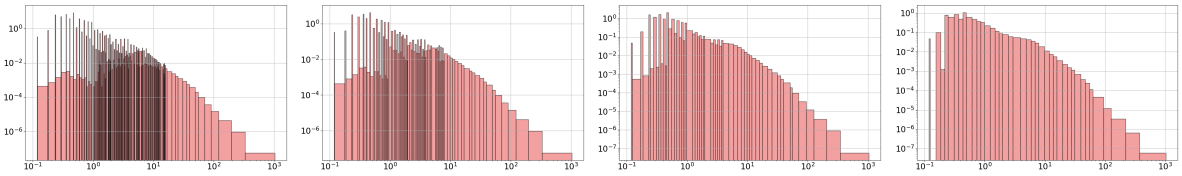
Applying the methodology introduced in Section 7.3, the indicators related to coarser-grained histograms are displayed in Figure 18. They suggest using the 6th histogram, obtained at depth 5, which is reduced to a unimodal distribution summarized with 23 intervals (see Figure 19b). Even with this small number of intervals, the floating-point representation allows to recover a global shape in power law. Still, the *information* curve in Figure 18 suggests a loss of information of more than 20% during this coarsening process, especially for crater diameters smaller than 10 km, as shown in Figure 20. This is consistent with the data collection process and may caution the data analyst to avoid drawing too precise conclusions from this data set.



(a) 1st interpretable histogram. (b) 6th interpretable histogram.

Figure 19. Moon crater Salamuniccar database.

[2] https://astrogeology.usgs.gov/search/map/Moon/Research/Craters/GoranSalamuniccar_MoonCraters



(a) 2nd (b) 3rd (c) 4th (d) 5th

Figure 20. Moon crater Salamuniccar database: intermediate histograms.

8.3. Moon crater Robbins database

The moon crater Robbins database [22] [3] contains approximately 1.3 million lunar impact craters. This recent database is estimated to be a complete census of all craters larger than approximately 1 to 2 km. The G-Enum-fp method directly builds an accurate and smooth histogram with 87 intervals that looks easy to interpret, as shown in Figure 21. It captures a power law decrease of the densities for craters between 1 km and 2,500 km, in line with astrophysics literature: power or multiple power laws are often used to fit the crater size distribution [23].

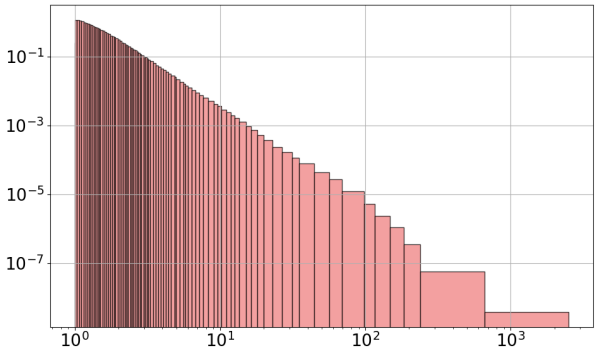


Figure 21. Moon crater Robbins database.

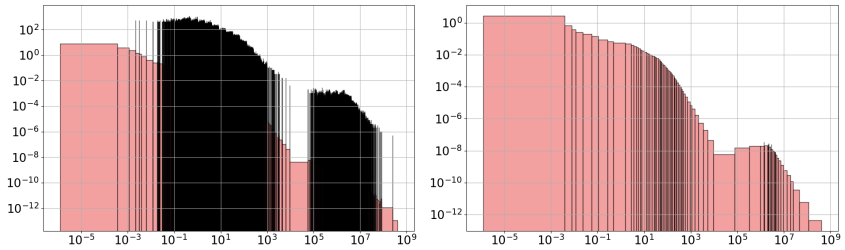
8.4. HYG stellar database

The HYG stellar database [22] [4] contains 119,614 star records from three catalogs: Hipparcos, Yale Bright Star and Gliese. We exploit this database to analyze the distribution of stars' luminosity, as multiples of the Solar luminosity. The raw histogram output by the G-Enum-fp method contains 17,032 intervals, including 8,513 spikes and 7,348 singularities (see Figure 22a). Although the luminosity is recorded with an accuracy of 10 decimal digits, there are only 13,451 distinct values for 119,614 records, which explains the raw histogram with a very comb-like shape.

No truncation gap is identified, and the 1st "interpretable" histogram obtained by the SRH method contains 81 intervals (see Figure 22b). This histogram, which spans over 14 orders of magnitude, benefits

[3] https://astrogeology.usgs.gov/search/map/Moon/Research/Craters/lunar_crater_database_robbins_2018

[4] https://www.datastro.eu/explore/dataset/hyg-stellar-database/information/



(a) Raw histogram. (b) 1st interpretable histogram.

Figure 22. HYG stellar database.

from the floating-point representation of the G-Enum-fp method. It highlights an interesting smooth and precise distribution of the luminosity of the stars, with an approximate mixture of decreasing power laws. This overall irregular shape of the histogram probably comes from the data set, which is a mixture of three rather different catalogs. For example, the Yale Bright Star catalog contains essentially all stars visible with the naked eye, which may explain the "bump" at the end of the histogram. And conversely, the Gliese catalog is the most comprehensive catalog of nearby stars, that contains many fainter stars not found in Hipparcos.

8.5. Orange call detail records

The data set studied here is composed of nearly 25 million entries of cumulated call durations for incoming calls collected during one day, for a large sample of phone numbers of the Orange telecommunication company[5]. The TMH method correctly identifies a truncation gap of 1 related to a one second precision and directly outputs the histogram displayed in Figure23, that consists of 222 intervals, with no singularity.

The fairly compact representation provided by this histogram summarizes a lot of information, beyond the overall shape of the distribution previously known as being heavy-tailed. For example, the first dense interval corresponds to phone numbers without any calls during the day, resulting in a cumulated call duration of 0 seconds. It spans over just one second but represents about half of the data set. Strikingly, the last interval covers about 3 million seconds and only accounts for 3 data entries, related to incoming calls collected one day that lasted more than one month. Another finding is that the heavy-tailed form exhibits two distinct power-law regimes, with a transition for durations of around one hour.

While the histogram looks globally smooth, there still are several dense peaks in-between. One could apply the methodology introduced in Section 7.3 to treat these peaks as digitization patterns and discard them, but they seem to be robust patterns that can only be eliminated after coarsening the histogram almost ten times. The three most notable peaks in the right (in the range $[10^3; 10^4]$ of Figure 23) correspond to call durations of exactly 1800, 3600 and 7200 seconds, that is 1/2, 1 and 2 hours. The smaller peaks at the beginning, which are less contrasted with the overall distribution, correspond to cumulated call durations of exactly $1, 2, 3, \ldots, 15$ minutes, plus $18, 19, 20$ minutes. A possible explanation for these denser peaks at round times might be relative to telecommunication services with a fixed contractual time, such as teleconferences.

[5]For privacy reasons, this data set is not publicly available.

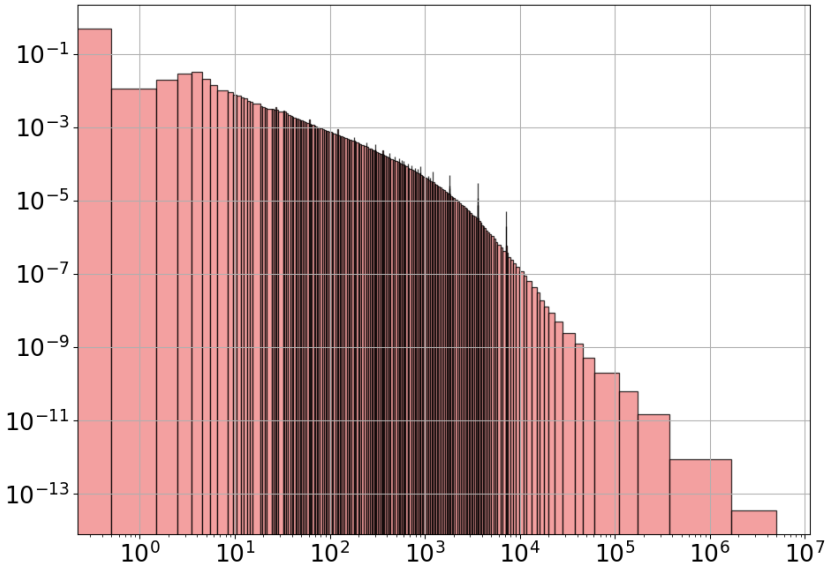


Figure 23. Orange CDR.

Overall, this histogram provides an insightful summary with both global and local patterns that were previously unknown to domain experts.

8.6. Web graph

The eu-2015 data set[6] is a large snapshot of the Web graph for European countries in 2015, collected by the Laboratory for Web Algorithmics [24, 25]. It consists of about 1 billion nodes and 92 billion edges. We focus on the in-degrees per node, that is the number of edges that point to a given node. These values range from 1 to more than 20 million, with 86 on average. There are only about 71,000 distinct values for in-degrees, which is a surprisingly very small number given the billion data entries. The visualization plot proposed on the website for the data set is shown in Figure 25a. It shows a frequency plot of the in-degrees, as well a smooth approximation of the distribution using Fibonacci binning.

The values of the in-degrees are integers, which is correctly retrieved by the TMH method, with a truncation gap of 1. The 1st obtained interpretable histogram, presented in Figure 25b, contains 15,034 intervals. The histogram shows a clear decreasing power law behavior, with most of the nodes having a very small in-degree, and very few nodes having huge in-degrees. The first interval contains about 230 million nodes with a in-degree of 1. The last interval contains only 7 nodes (among 1 billion), for in-degrees spanning from 2 million to 20 million.

Although the general shape of the histogram is quite straightforward, it looks noisy for in-degrees between 1,000 and 1,000,000. A close inspection at some of these dense peaks shows that they are not pure noise: they actually reveal some surprising patterns. For example, there is one interval that contains 59 nodes with 297,690 or 297,691 in-degrees. It is surrounded by two intervals that are about one thousand times less dense: one with 14 nodes spanning over 2000 in-degrees values and another

[6] Available at http://law.di.unimi.it/webdata/eu-2015/ with some visualization plots

with 152 nodes spanning over 15,000 in-degrees values. Having such a concentration of node with almost exactly the same huge in-degree might be the signature of a Web farm.

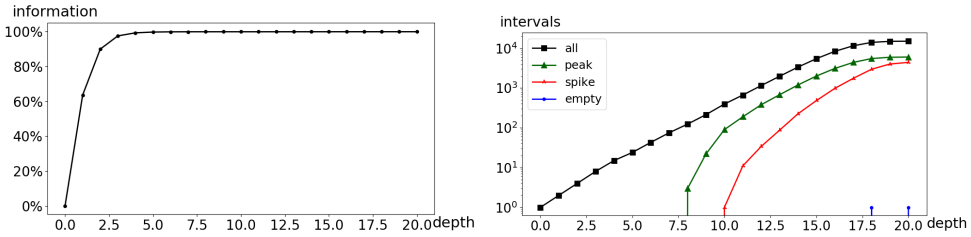
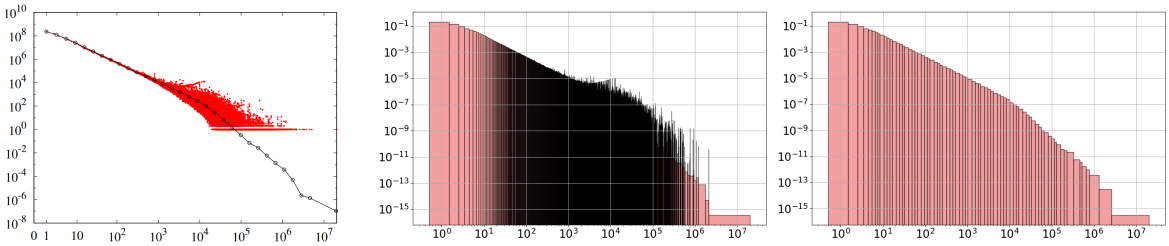


Figure 24. LabWeb: indicators per depth of granularity.

Applying the methodology introduced in Section 7.3 allows to get a simplified summary of the in-degrees distribution and to capture its global shape. The indicators displayed in Figure 24 suggest to coarsen the histogram from its initial granularity depth 20 down to 7, which is a considerable coarsening albeit with minimal loss of information. The simplified histogram, shown in Figure 25c, consists of 75 intervals chosen among only 93 floating-point bins. It has a smooth decreasing power-law shape on seven orders of magnitude for the in-degrees and 15 orders of magnitude for the densities. The density estimation is accurate enough to distinguish two regimes for the power law, before and after around 10,000 in-degrees. It should be noted that these results are visually consistent with the frequency plot and Fibonacci binning provided with the data set (cf. Figure 25a), both for the noisy patterns for in-degrees above 1,000 and for the overall shape of the distribution.



(a) frequency plot (b) 1st interpretable histogram. (c) 14th interpretable histogram.

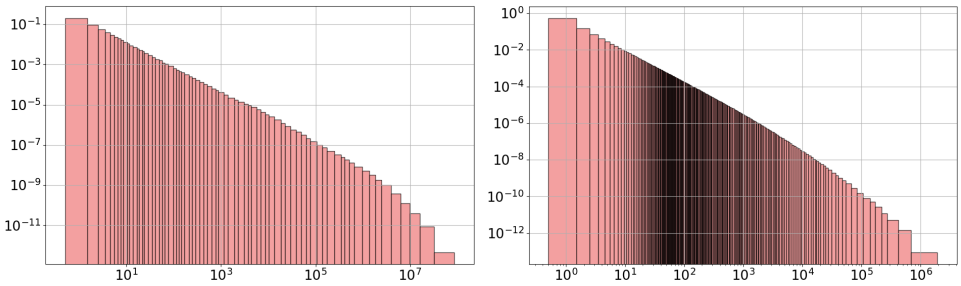
Figure 25. LabWeb.

8.7. *New York Times Annotated Corpus*

The New York Times Annotated Corpus (NYTAC)[7] contains over 1.8 million articles written and published by the New York Times between January 1, 1987 and June 19, 2007. Texts can be decomposed into sets of tokens, where tokens can be words, letters or even bytes, or their sequences named n-grams: n-grams of words, letters or bytes. In this section, we study the decomposition of the articles of the NYTAC into n-grams of bytes. There are about 166,000 distinct 3-grams in the corpus and their

[7] Available from the Linguistic Data Consortium (LDC) at https://catalog.ldc.upenn.edu/LDC2008T19

total number of occurrences is about 6.2 billion. Let us call the *popularity* of a 3-gram its number of occurrences in the corpus. A *popular* 3-gram has a large number of occurrences, while a *noteless* one has a small number of occurrences[8]. We apply the G-Enum-fp method to summarize the density of this popularity variable for the 3-grams in the NYTAC. The popularity is an integer variable, which is recovered by the TMH method using a truncation gap of 1. The resulting histogram has 73 intervals and is directly interpretable, with a very smooth shape, as shown in Figure 26a.



(a) Histogram of 3-grams' popularity. (b) Histogram of 9-grams' popularity.

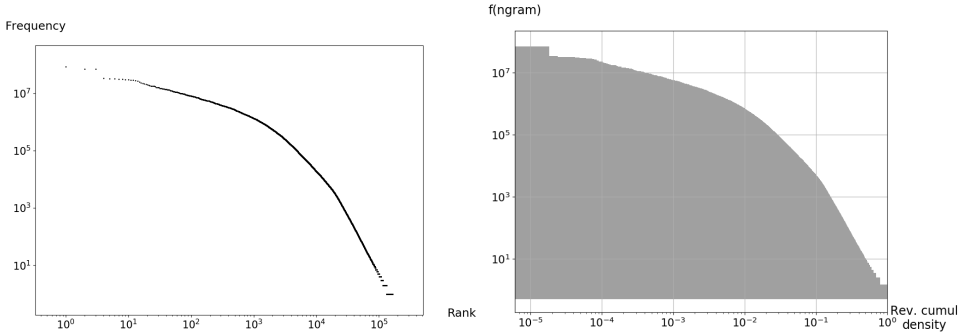
Figure 26. New York Times Annotated Corpus: popularity of n-grams.

We evaluated the popularity of all n-grams of bytes, from 1-grams to 9-grams. We report the results for the 9-grams, which are the most numerous, with about 254 million distinct 9-grams in the corpus. The noteless 9-grams are very frequent, with over 134 million 9-grams having a popularity of 1. And the popular 9-grams are rare, the most popular *" for the "* having a popularity of around 1.9 million. The resulting histogram displayed in Figure 26b is very accurate, with 230 intervals and a density spanning over 12 orders of magnitude. The length of the 65 first intervals is 1 while that of the last interval is greater than 1.2 million. This last interval contains the 27 most popular 9-grams, that behave as outliers in this histogram.

The histograms build for each kind the n-grams provide a smooth and accurate estimation of the underlying probability density function of the popularity of the n-grams. Although the data set combines several challenges, with integer data in addition to heavy-tailed distribution and large size, the retrieved histograms do not suffer from com-shaped or noisy patterns such as those displayed in Figure 15. They are remarkably smooth and parsimonious, which make them easy to analyze and interpret. The shape of the density function given by the histograms is very similar for each kind of n-grams. It is almost a straight line, especially for the noteless n-grams, but is still concave, especially for the popular n-grams.

The frequency of tokens in text corpora has been widely studies in the literature. The Zipf's law [26] for word tokens in a corpus states that if f is the frequency of a word in the corpus and r is the rank of a word by decreasing frequency, then $f = k/r$ where k is a constant for the corpus. When f is drawn in relation to r using a $\log \times \log$ graph, which is called a Zipf curve, a straight line is obtained with a slope of -1. To take into account deviations from this behavior, several modifications of the law have been proposed, in particular the one derived theoretically in [27], $f = \frac{k}{(r+\alpha)^\beta}$ where α and β are constants for the analyzed corpus.

[8]This notion of *popularity* for 3-grams has been introduced to avoid confusing comments such as "frequent words are rather infrequent" or "rare words are very frequent".



(a) Empirical Zipf curve. (b) Estimation using a histogram.

Figure 27. New York Times Annotated Corpus: Zipf curves of 3-grams.

The Zipf curve for 3-grams in the NYTAC is drawn in Figure 27a. Except for the least frequent 3-gram on the right side of the curve, the shape of the curve is clearly concave, far from a straight line. We now focus on the relationship between the Zipf curve for 3-grams in Figure 27a and the probability density function of the popularity of 3-grams in Figure 26a. Let $w_i, 1 \leqslant i \leqslant n$ be a 3-grams, $f(w_i)$ its frequency in the corpus, and $r(w_i)$ its rank by decreasing frequency. The Zipf curve plots $r(w_i)$ versus $f(w_i)$ using a $\log \times \log$ scale. Let us now focus on the complementary cumulative distribution function (ccdf) of the popularity of 3-grams, that is

$$\bar{F}_X(x) = P(X > x) \tag{5}$$

where X is the popularity variable, which is estimated using the frequency of the 3-grams. The empirical ccdf is computed from the data entries in the sample according to

$$\hat{F}_X(x) = \frac{\text{number of 3-grams in the corpus with popularity above x}}{n}, \tag{6}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{(f(w_i)>x)}, \tag{7}$$

where $\mathbb{1}_{(f(w_i)>x)}$ is 1 if $f(w_i) > x$ and 0 otherwise. When 3-grams are sorted by decreasing frequencies, we have

$$\hat{F}_X(f(w_i)) = \frac{1}{n} r(w_i). \tag{8}$$

This demonstrates that the Zipf curve is none other than the empirical ccdf of the popularity drawn using a $\log \times \log$ scale with the two axes inverted [28]. As the histogram displayed in Figure 26a represents an estimate of the popularity distribution function of the 3-grams, we exploit it to get an estimate of the ccdf and obtain the graph displayed in Figure 27b, based on 73 intervals instead of 166,000 points in the Zipf curve of Figure 27a. This shows that the histograms obtained using the G-Enum-fp method provide a very accurate summary of the Zipf curve, much easier to study given their parsimony. It opens new

research avenues for the study of the word frequency distribution, either directly using the probability distribution or its cumulative variants[9].

9. Conclusion

In line with our goal of exploratory analysis of large scale real-world data sets, we chose the G-Enum histogram method [6] as our starting point because it is automatic, scalable, parsimonious and achieves state-of-the art accuracy in density estimation. The G-Enum method builds irregular histograms with intervals of variable lengths, based on a modeling space consisting of equal-width elementary bins. This makes sense for a piecewise-constant density estimator for distributions with real values in $\mathbb{R}$. Although it works well in many cases, this method cannot cope with distant outliers or heavy-tailed distributions. When the number of equal-width elementary bins required to cover the entire value domain increases to its limits, the elementary bins become over-lengthy to properly approximate the dense density regions of the underlying distributions.

In this paper, we have suggested to extend the G-Enum method, by replacing its equal-width elementary bins by floating-point elementary bins that exploit the floating-point representation of real numbers on computers. This alternative representation space enables to treat any data set that can be represented on a computer, rather than data sets with values in $\mathbb{R}$. The new method, named G-Enum-fp, keeps most of the features of the G-Enum method: the modeling space (only the definition of the elementary bins has changed), the evaluation criterion and the optimization heuristics. This allows to inherit from the appealing properties of the G-Enum method: parameter-less, robustness, accuracy, parsimony and scalability. Extensive experiments have been conducted to analyze the impact of this new representation space using various artificial distributions. The results show indistinguishable performance in the case of standard regular distributions. On the other hand, in the case of outliers or heavy-tailed distribution, the G-Enum-fp method brings considerable improvements, as it can accurately approximate the underlying distributions regardless of their shape and scale.

However, these very promising results collapsed during the first experiments with real data. The problem is not the limited precision of the computer representation of real numbers. As shown by artificial experiments with known distributions, the precision of the 15-digit decimal mantissa is clearly sufficient, even in the case of very large data sets. The problem is that real-world data sets involve a digitization process, with many potential issues such as for example, inherently integer data, biased data collection, limited recording accuracy, data transformation, rounding or truncation errors. As the size of the data sets increases, the digitized structure of the data may dominate the statistical structure of the original data, and accurate histograms retrieve the dominant structure, which may be of little interest for exploratory analysis. Abandoning the objective of fully automated histograms for exploratory analysis, we have proposed heuristics to process truncated data and eliminate singularities, as well as a methodology to facilitate the task of recovering the statistical structure of the data beyond their digitization structure. Extensive experiments on large scale real-world data sets show the effectiveness of the approach.

The first immediate objective of future work is to apply the methodology proposed in this paper to discover new insights in areas where histograms could not be easily applied previously. Another research direction includes extensions of the G-Enum-fp method to the processing of huge data stores or fast data streams. Finally, it is noteworthy that one of the most striking surprises in this article is the

[9]The normalized word frequency $f(w_i)/\sum_j f(w_j)$ could also be used to obtain less corpus-dependent information.

strong interweaving of digitization and statistical structures in large real-world data sets. This implies that accurate methods can sometimes produce results without much interest. As the current trend in supervised classification is to apply methods with numerous parameters on very large data sets, one can cautiously consider the excellent accuracy of the predictions obtained. Analyzing this potential issue appears to be a promising direction for research.

References

[1] J. Rissanen, T.P. Speed and B. Yu, Density estimation by stochastic complexity, *IEEE Transactions on Information Theory* **38**(2) (1992), 315–323.

[2] P. Kontkanen and P. Myllymäki, MDL Histogram Density Estimation, in: *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, M. Meila and X. Shen, eds, Proceedings of Machine Learning Research, Vol. 2, PMLR, 2007, pp. 219–226.

[3] P.L. Davies and A. Kovac, Densities, spectral densities and modality, *Ann. Statist.* **32**(3) (2004), 1093–1136. doi:10.1214/009053604000000364.

[4] Y. Rozenholc, T. Mildenberger and U. Gather, Combining regular and irregular histograms by penalized likelihood, *Computational Statistics and Data Analysis* **54**(12) (2010), 3313–3323. doi:https://doi.org/10.1016/j.csda.2010.04.021. http://www.sciencedirect.com/science/article/pii/S0167947310001660.

[5] J.D. Scargle, J.P. Norris, B. Jackson and J. Chiang, STUDIES IN ASTRONOMICAL TIME SERIES ANALYSIS. VI. BAYESIAN BLOCK REPRESENTATIONS, *The Astrophysical Journal* **764**(2) (2013), 167. doi:10.1088/0004-637x/764/2/167. http://dx.doi.org/10.1088/0004-637X/764/2/167.

[6] V. Zelaya Mendizábal, M. Boullé and F. Rossi, Fast and fully-automated histograms for large-scale data sets, *Computational Statistics & Data Analysis* **180** (2023), 107668. doi:https://doi.org/10.1016/j.csda.2022.107668. https://www.sciencedirect.com/science/article/pii/S0167947322002481.

[7] J. Rissanen, A Universal Prior for Integers and Estimation by Minimum Description Length, *Ann. Statist.* **11**(2) (1983), 416–431. doi:10.1214/aos/1176346150.

[8] M. Boullé, F. Clérot and C. Hue, Revisiting enumerative two-part crude MDL for Bernoulli and multinomial distributions (Extended version), *arXiv* **1608.05522** (2016).

[9] L. Davies, U. Gather, D. Nordman and H. Weinert, A comparison of automatic histogram constructions, *ESAIM: Probability and Statistics* **13** (2009), 181–19.

[10] D. Freedman and P. Diaconis, On the histogram as a density estimator:L2 theory, *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* **57**(4) (1981), 453–476. doi:10.1007/BF01025868.

[11] V. Hodge and J. Austin, A survey of outlier detection methodologies, *Artificial intelligence review* **22** (2004), 85–126.

[12] J. Zhang, Advancements of outlier detection: A survey, *ICST Transactions on Scalable Information Systems* **13**(1) (2013), 1–26.

[13] M. Gebski and R.K. Wong, An efficient histogram method for outlier detection, in: *Advances in Databases: Concepts, Systems and Applications: 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007. Proceedings 12*, Springer, 2007, pp. 176–187.

[14] M. Boullé, Two-level histograms for dealing with outliers and heavy tail distributions, *arXiv* **2306.05786** (2023).

[15] IEEE, IEEE Standard for Binary Floating-Point Arithmetic, *ANSI/IEEE Std 754-1985* (1985), 1–20. doi:10.1109/IEEESTD.1985.82928.

[16] D. Dua and C. Graff, UCI Machine Learning Repository, 2017. http://archive.ics.uci.edu/ml.

[17] K.H. Knuth, J.P. Castle and K.R. Wheeler, Identifying excessively rounded or truncated data, in: *Compstat 2006 - Proceedings in Computational Statistics*, A. Rizzi and M. Vichi, eds, Springer, 2006, pp. 313–323.

[18] A. Azzalini and A.W. Bowman, A Look at Some Data on the Old Faithful Geyser, *Journal of the Royal Statistical Society. Series C (Applied Statistics)* **39**(3) (1990), 357–365.

[19] A. Marx, L. Yang and M. van Leeuwen, Estimating Conditional Mutual Information for Discrete-Continuous Mixtures using Multi-Dimensional Adaptive Histograms, in: *Proceedings of the 2021 SIAM International Conference on Data Mining, SDM 2021, Virtual Event, April 29 - May 1, 2021*, C. Demeniconi and I. Davidson, eds, SIAM, 2021, pp. 387–395.

[20] J.W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, 1977.

[21] G. Salamunićcar, S. Lončarić and E. Mazarico, LU60645GT and MA132843GT catalogues of Lunar and Martian impact craters developed using a Crater Shape-based interpolation crater detection algorithm for topography data, *Planetary and Space Science* **60**(1) (2012), 236–247.

[22] S.J. Robbins, A New Global Database of Lunar Impact Craters >1-2 km: 1. Crater Locations and Sizes, Comparisons With Published Databases, and Global Analysis, *Journal of Geophysical Research (Planets)* **124**(4) (2019), 871–892. doi:10.1029/2018JE005592.
[23] N. Wang and J.-L. Zhou, Determining proportions of lunar crater populations by fitting crater size distribution, *Research in Astronomy and Astrophysics* **16**(12) (2016), 185.
[24] P. Boldi and S. Vigna, The WebGraph Framework I: Compression Techniques, in: *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, ACM Press, 2004, pp. 595–601.
[25] P. Boldi, M. Rosa, M. Santini and S. Vigna, Layered Label Propagation: A MultiResolution Coordinate-Free Ordering for Compressing Social Networks, in: *Proceedings of the 20th international conference on World Wide Web*, S. Srinivasan, K. Ramamritham, A. Kumar, M.P. Ravindra, E. Bertino and R. Kumar, eds, ACM Press, 2011, pp. 587–596.
[26] G.K. Zipf, *Human Behaviour and the Principle of Least Effort*, Addison-Wesley, 1949.
[27] B. Mandelbrot, An Information Theory of the Statistical Structure of Language, in: *Communication Theory*, Academic Press, 1953, pp. 486–502.
[28] M. Newman, Power laws, Pareto distributions and Zipf's law, *Contemporary Physics* **46**(5) (2005), 323–351. doi:10.1080/00107510500052444.
[29] C.E. Shannon, A mathematical theory of communication, Technical Report, 27, Bell systems technical journal, 1948.
[30] P.D. Grünwald, *The minimum description length principle*, Adaptive computation and machine learning, MIT Press, 2007.
[31] P. Kontkanen, *Computationally efficient methods for MDL-optimal density estimation and data clustering*, Department of Computer Science, series of publications A, report, 2009-11, University of Helsinki, 2009.
[32] T. Mononen and P. Myllymäki, Computing the multinomial stochastic complexity in sub-linear time, in: *Proceedings of the 4th European Workshop on Probabilistic Graphical Models (PGM-08), September 17-19, 2008, Hirtshals, Denmark*, 2008, pp. 209–216, Volume: Proceeding volume:.
[33] J. Rissanen, Fisher information and stochastic complexity., *IEEE Transactions on Information Theory* **42**(1) (1996), 40–47.
[34] W. Szpankowski, On Asymptotics Of Certain Recurrences Arising In Universal Coding, *Problems of Information Transmission* **34**(2) (1998), 142–146.
[35] C. Brooks, E.A. Lee, X. Liu, S. Neuendorffer, Y. Zhao and H. Zheng, Heterogeneous Concurrent Modeling and Design in Java, Technical Report, Technical Memorandum UCB/ERL M04/27, University of California, 2004. http://ptolemy.eecs.berkeley.edu/publications/papers/04/ptIIDesignIntro/.
[36] M. Boullé, Recherche d'une représentation des données efficace pour la fouille des grandes bases de données, PhD thesis, Ecole Nationale Supérieure des Télécommunications, 2007.
[37] M. Boullé, MODL: a Bayes optimal discretization method for continuous attributes, *Machine Learning* **65**(1) (2006), 131–165.
[38] J.J. Rissanen, Fisher information and stochastic complexity, *IEEE Transactions on Information Theory* **42**(1) (1996), 40–47. doi:10.1109/18.481776.
[39] D. Dua and C. Graff, UCI Machine Learning Repository, 2017. http://archive.ics.uci.edu/ml.
[40] P. Kontkanen, W.L. Buntine, P. Myllymäki, J. Rissanen and H. Tirri, Efficient Computing of Stochastic Complexity, in: *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003, Key West, Florida, USA, January 3-6, 2003*, 2003. http://research.microsoft.com/en-us/um/cambridge/events/aistats2003/proceedings/172.pdf.
[41] J. Rissanen, Modeling by shortest data description, *Automatica* **14** (1978), 465–471.
[42] L. Davies and A. Kovac, *ftnonpar: Features and Strings for Nonparametric Regression*, 2012, R package version 0.1-88. https://CRAN.R-project.org/package=ftnonpar.
[43] T. Mildenberger, Y. Rozenholc and D. Zasada, *histogram: Construction of Regular and Irregular Histograms with Different Options for Automatic Choice of Bins*, 2019, R package version 0.0-25. https://CRAN.R-project.org/package=histogram.
[44] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T.E. Oliphant, Array programming with NumPy, *Nature* **585**(7825) (2020), 357–362. doi:10.1038/s41586-020-2649-2.
[45] Astropy Collaboration, A.M. Price-Whelan, P.L. Lim, N. Earl, N. Starkman, L. Bradley, D.L. Shupe, A.A. Patil, L. Corrales, C.E. Brasseur, M. N"othe, A. Donath, E. Tollerud, B.M. Morris, A. Ginsburg, E. Vaher, B.A. Weaver, J. Tocknell, W. Jamieson, M.H. van Kerkwijk, T.P. Robitaille, B. Merry, M. Bachetti, H.M. G"unther, T.L. Aldcroft, J.A. Alvarado-Montes, A.M. Archibald, A. B'odi, S. Bapat, G. Barentsen, J. Baz'an, M. Biswas, M. Boquien, D.J. Burke, D. Cara, M. Cara, K.E. Conroy, S. Conseil, M.W. Craig, R.M. Cross, K.L. Cruz, F. D'Eugenio, N. Dencheva, H.A.R. Devillepoix, J.P. Dietrich, A.D. Eigenbrot, T. Erben, L. Ferreira, D. Foreman-Mackey, R. Fox, N. Freij, S. Garg, R. Geda, L. Glattly, Y. Gondhalekar, K.D. Gordon, D. Grant, P. Greenfield, A.M. Groener, S. Guest, S. Gurovich, R. Handberg, A. Hart, Z. Hatfield-Dodds, D. Homeier, G. Hosseinzadeh, T. Jenness, C.K. Jones, P. Joseph, J.B. Kalmbach,

E. Karamehmetoglu, M. Kaluszy'nski, M.S.P. Kelley, N. Kern, W.E. Kerzendorf, E.W. Koch, S. Kulumani, A. Lee, C. Ly, Z. Ma, C. MacBride, J.M. Maljaars, D. Muna, N.A. Murphy, H. Norman, R. O'Steen, K.A. Oman, C. Pacifici, S. Pascual, J. Pascual-Granado, R.R. Patil, G.I. Perren, T.E. Pickering, T. Rastogi, B.R. Roulston, D.F. Ryan, E.S. Rykoff, J. Sabater, P. Sakurikar, J. Salgado, A. Sanghi, N. Saunders, V. Savchenko, L. Schwardt, M. Seifert-Eckert, A.Y. Shih, A.S. Jain, G. Shukla, J. Sick, C. Simpson, S. Singanamalla, L.P. Singer, J. Singhal, M. Sinha, B.M. SipHocz, L.R. Spitler, D. Stansby, O. Streicher, J. Sumak, J.D. Swinbank, D.S. Taranu, N. Tewary, G.R. Tremblay, M.d. Val-Borro, S.J. Van Kooten, Z. Vasovi'c, S. Verma, J.V. de Miranda Cardoso, P.K.G. Williams, T.J. Wilson, B. Winkel, W.M. Wood-Vasey, R. Xue, P. Yoachim, C. Zhang, A. Zonca and Astropy Project Contributors, The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package, *apj* **935**(2) (2022), 167. doi:10.3847/1538-4357/ac7c74.

[46] M. Boullé, Two-level histograms for dealing with outliers and heavy tail distributions, 2023.

Appendix A. G-Enum-fp histogram method

In the Section 4, we have exploited the floating-point representation of real values to introduce an alternative definition of the elementary bins used as building blocks for a new histogram method called **G-Enum-fp**. We first summarize the principles of this new method, then detail its specific components.

A.1. Principle

The G-Enum method exploits a representation space based on elementary equal-width bins, using a granularity parameter G to explore simplified versions of this representation space. It exploits these elementary bins as building blocks that provide a set of predefined bounds, from which the bounds of the histogram intervals are chosen.

The main novelty of the G-Enum-fp method is to replace the elementary equal-width bins of G-Enum with the floating-point bins and their granulated hierarchy introduced in Section 4. In addition to this major change, the G-Enum-fp method extends the modeling space and the optimization algorithms of G-Enum to take full account of the particularities of floating-point bins. With, the G-Enum-fp method, the bounds of the whole numeric domain are explicitly specified as hyper-parameters, in Section A.2. Another difference related to floating-point representation is the management of the singularity around 0, which relies on a central bin. This leads to the introduction of a new parameter in Section A.3. Once the domain's bounds and the central bin are specified, the G-Enum-fp method extends the G-Enum method, as detailed in Section A.4.

A.2. Specification of domain bounds

Domain bounds are usually derived from the data in most existing histogram methods. This sometimes results in some undesirable side-effects, such as under or over-estimated density around the extrema values. While the impact is negligible when density estimation is the only objective, this may cause some harm in the context of exploratory analysis. To reduce such potential problems, we suggest to introduce hyper-parameters to explicitly specify the bounds of the numerical domain. Our objective is also to clarify some modeling choices and to exploit a full Bayesian model selection approach.

With the G-Enum-fp method, we explicitly specify the lower and upper bounds D_{lb}, D_{ub} of the numerical domain by choosing two extrema floating-points bins, as well as a central bin that is compatible with the data set[10]. To do this, we introduce the following hyper-parameters:

- i_{lb}: exponent of the main bin $B_{M_{lb}}$ containing the lower bound D_{lb},
- i_{ub}: exponent of the main bin $B_{M_{ub}}$ containing the upper bound D_{ub},
- i_{cb}: exponent of the central bin that is empty, except for the value zero,
- d_b: digit precision used to encode the index of the mantissa bins containing the lower and upper bounds,
- m_{lb}, m_{ub}: index of the mantissa bins $B_{m_{lb}}, B_{m_{ub}}$ with d_b digits precision, containing the lower and upper bounds within their main bins.

Our objective is to set the values of these hyper-parameters once for all, before optimizing the histograms. To do this, we assume that we only know the number of instances n, the extreme values x_{min},

[10] In Section 4, the floating-point bins were introduced using directly the extreme values of the data set as the bounds of the value domain, only for reasons of simplicity.

x_{max} and the smallest non-zero value in the data set. With no further assumption about the distribution of the data, we optimize the hyper-parameters using a Bayesian approach, more precisely a Bayesian interpretation of MDL criteria, since negative logarithms of probabilities can be considered as coding lengths [29].

Prior. For each main bin $B_{M_{lb}}, B_{M_{lb}}$, we use 3 bits to encode the choice between central or exponent bin, the sign of the bin and the sign of the exponent. To introduce a slight bias towards small exponents, we use the universal prior for integers of Rissanen [7] to encode the absolute value of the exponent. This results in

$$3\log 2 + \log^*(1 + |i_{lb}|) + 3\log 2 + \log^*(1 + |i_{ub}|). \tag{9}$$

The sign and the absolute value of the exponent i_{cb} are encoded in the same way, only if necessary, that is if it cannot be deduced from $B_{M_{lb}}, B_{M_{ub}}$:

$$\mathbb{1}_{\{D_{lb} \leqslant 0 < D_{ub}\}}(\log 2 + \log^*(1 + |i_{cb}|)). \tag{10}$$

We exploit again the universal prior for integers of Rissanen to encode the digit precision d_b, and then d_b bits are necessary to encode the index of the mantissa bins $B_{m_{lb}}, B_{m_{ub}}$ which extremities are D_{lb} and D_{ub}. We get

$$\log^*(1 + d_b) + 2d_b \log 2. \tag{11}$$

Altogether, the criterion for encoding the prior terms of the hyper-parameters is

$$cost_{HP_P} = (6 + 2d_b)\log 2 \tag{12}$$

$$+ \log^*(1 + |i_{lb}|) + \log^*(1 + |i_{ub}|) + \log^*(1 + d_b) \tag{13}$$

$$+ \mathbb{1}_{\{D_{lb} \leqslant 0 < D_{ub}\}}(\log 2 + \log^*(1 + |i_{cb}|)). \tag{14}$$

As $\log^* \approx \log$, this gives the approximation $cost_{HP_P} = 2d_b \log 2 + O(\log(1 + d_b))$, which shows that the prior terms mainly increase linearly with the mantissa precision used to encode the domain bounds.

Likelihood. Given i_{lb}, i_{ub}, i_{cb}, the number n_B of main bins can be deduced, as well as the maximum number of bits $d_{b_{max}}, 0 \leqslant d_{b_{max}} \leqslant d_{max} = 30$, available to encode the mantissa of the domain bounds: $d_{b_{max}} = d_{max} - \lceil \log_2 n_B \rceil$.

As with the G-Enum method, we encode the position of each instance of the data set on the value domain using the *bin index terms* (see Table 1). We only know the extreme values x_{min}, x_{max} and the smallest non-zero value in the data set, and have no assumption about the distribution of the data. We then exploit only one interval, and the bin index terms reduce to

$$n \log E, \tag{15}$$

where E is total number of elementary ϵ-bins in the value domain. Let ϵ be the minimum length over all the elementary floating-point bins of the value domain, that is the length of the mantissa bins within the

main bins having the smallest exponent when the maximum granularity is chosen. All the elementary floating-point bins have a length that a multiple of ϵ by a power of two factor, and E can be obtained using $E = (D_{ub} - D_{lb})/\epsilon$, resulting in the following likelihood terms

$$cost_{HP_L} = n \log \frac{D_{ub} - D_{lb}}{\epsilon}, \tag{16}$$

that increases with the range $(D_{ub} - D_{lb})$ of the domain.

Optimization. To start with a minimal range $(D_{ub} - D_{lb})$, we first choose the exponent of the main bins of the domain from the extreme values of the data set according to:

- i_{lb}: exponent of the main bin containing the minimum value x_{min} of the data set,
- i_{ub}: exponent of the main bin containing the maximum value x_{max} of the data set,
- i_{cb}: exponent of the largest central bin that does not contain any non-zero value of the data set.

We now have to optimize the mantissa precision $d_b, 0 \leqslant d_b \leqslant d_{b_{max}}$, used to encode the domain bounds D_{lb} and D_{ub}. Removing the fixed terms from $cost_{HP_P} + cost_{HP_L}$, this reduces to optimizing

$$\log(1 + d_b) + 2d_b \log 2 + n \log(D_{ub}(d_b) - D_{lb}(d_b)), \tag{17}$$

where $D_{lb}(d_b)$ and $D_{ub}(d_b)$ correspond to the encoding of x_{min} and x_{max} using a floating-point representation with d_b bits of mantissa precision.

We obtain

$$d_b = \underset{0 \leqslant d \leqslant d_{b_{max}}}{\operatorname{argmin}} \log(1 + d) + 2d \log 2 + n \log(D_{ub}(d) - D_{lb}(d)). \tag{18}$$

Illustration. Let us consider a data set with $n = 1,000$ instances in $[10.4, 114.7]$. The index of the main bin containing the domain lower bound is $i_{lb} = 3$, as $x_{min} = 10.4 \in]2^3, 2^4]$. For the domain upper bound, we have $i_{ub} = 6$, as $x_{max} = 114.7 \in]2^6, 2^7]$. As all the values of the data set are strictly positive, we do not need a central bin. The domain is covered by $n_b = 5$ main bins, and we obtain $d_{b_{max}} = d_{max} - \lceil \log_2 n_B \rceil = 27$. Table 3 shows the optimization details for 0 to 15 bits precision of the mantissa bins. With 0 bits, the domain bounds are coarse since they reduce to the bounds of their main bins. With 15 bits, they are very precise and very close to the extreme values of the data set. The best trade-of between the number of bits need to encode the domain bounds and the likelihood of the data is achieved with 6 bits, as highlighted in Table 3.

A.3. Choice of the central bin

We introduce a new parameter i_{cen} to choose the exponent of the central bin used to obtain a representation space based on floating-point bins. As a matter of fact, we can choose any value of i_{cen} compliant with the domain bounds introduced in Section A.2, that is between $i_{cen_{min}} = i_{cb}$ and $i_{cen_{max}} = 1 + max(i_{lb}, i_{ub})$.

For example, in Figure 3, we have $i_{lb} = 1, i_{ub} = 2, i_{cb} = -3$, and the floating-point representation is chosen for $i_{cen_{min}} = i_{cb}$. At each granularity, we obtain a set of floating-point bins of varying sizes according to their main bins that exploit exponents between -3 and 3. Alternatively, we could choose $i_{cen_{min}} = 3$ to have the data contained in the two central bins $]-2^3, 0] \cup]0, 2^3]$ and get an equal-width

Table 3
Optimization of domain bounds

d	$D_{lb}(d)$	$D_{ub}(d)$	Total cost	$\log(1+d)$	$2d\log 2$	$n\log(D_{ub}(d)-D_{lb}(d))$
0	8	128	4792.9	2.6	2.8	4787.5
1	8	128	4794.8	3.1	4.2	4787.5
2	10	128	4779.9	3.7	5.5	4770.7
3	10	120	4711.5	4.1	6.9	4700.5
4	10	116	4676.2	4.4	8.3	4663.4
5	10.25	116	4675.5	4.7	9.7	4661.1
6	**10.375**	**115**	**4666.4**	**4.9**	**11.1**	**4650.4**
7	10.375	115	4668.0	5.1	12.5	4650.4
8	10.375	114.75	4667.1	5.3	13.9	4648.0
9	10.3906	114.75	4668.5	5.4	15.2	4647.8
10	10.3984	114.75	4670.0	5.6	16.6	4647.8
11	10.3984	114.719	4671.2	5.7	18.0	4647.5
12	10.3984	114.703	4672.5	5.8	19.4	4647.3
13	10.3994	114.703	4674.0	5.9	20.8	4647.3
14	10.3999	114.703	4675.5	6.0	22.2	4647.3
15	10.3999	114.701	4677.0	6.1	23.6	4647.3

representation across the entire numerical domain at each granularity. And any central bin exponent between the *maximally floating-point representation* and the *maximally equal-width representation* is possible.

Prior. We have $i_{cen} \in [i_{cen_{min}}, i_{cen_{max}}]$. An equal-width representation looks more suitable to the inference of histogram, as they are piece-wise constant density estimator. We translate this preference into a slight bias towards equal-width representation using the uniform prior of Rissanen to encode the exponent of the central bin:

$$\log^*(1 + i_{cen_{max}} - i_{cen}). \tag{19}$$

Optimization. To avoid a loop over all possible values of $i_{cen} \in [i_{cen_{min}}, i_{cen_{max}}]$, we start with the maximally floating-point representation and optimize a fist histogram $M(i_{cen_{min}})$ for $i_{cen} = i_{cen_{min}}$, using the optimization algorithm summarized in Section A.4. With this histogram, the accuracy of the interval boundaries is maximal around zero, as the values close to zero are located in the main bins with the smaller exponents.

For larger values of i_{cen} and keeping the same granularity, the central bins become larger, as well as their mantissa bins. Therefore, instances separated in adjacent intervals with the first histogram $M(i_{cen_{min}})$ may no longer be separable for larger values of i_{cen}. We then try to adapt the boundaries of each interval of $M(i_{cen_{min}})$ to the new representations induced by larger values of i_{cen}, and test whether the instances of the data set can still be divided into intervals of the same frequency. This test can be performed in $O(K)$, where K is the number of intervals of $M(i_{cen_{min}})$. By dichotomy, we look for the largest value $i_{cen_{opt}}$ of i_{cen} where this property is still valid. This dichotomous search can be performed in $O(K\log(i_{cen_{max}} - i_{cen_{min}} + 1)$ which is bounded by $O(n\log r_e)$ and is negligible in practice compared to the main optimization time.

We finally optimize a second histogram $M(i_{cen_{opt}})$ for $i_{cen} = i_{cen_{opt}}$, and keep the histogram having the best cost. This is summarized by the three-steps process below:

(1) optimize a first histogram $M(i_{cen_{min}})$ for $i_{cen} = i_{cen_{min}}$,

(2) by dichotomy in $[i_{cen_{min}}, i_{cen_{max}}]$, search for the largest value $i_{cen_{opt}}$ of i_{cen} related to a floating-point representation where the partition of the data set according to $M(i_{cen_{min}})$ is still possible while keeping the same frequencies per interval,
(3) optimize a second histogram $M(i_{cen_{opt}})$ for $i_{cen} = i_{cen_{opt}}$ and keep this histogram if its cost is better than that of the first histogram.

A.4. Main optimization algorithm

For fixed domain boundaries and exponent of the central bin, we obtain a set a floating-point bins for each granularity (see Section 4). The rest of the G-Enum-fp model is identical to the G-Enum model, as we have to choose the granularity, the number of intervals, the partition of the considered elementary bins into intervals and the interval frequencies. The only difference is that the elementary bins are floating-point bins rather that equal-width bins. We refer to Table 1 that summarizes the G-Enum criterion to comment the slight changes related to G-Enum-fp method.

Hyper-parameter terms. The hyper-parameter terms that encode the domain bounds (see Section A.2) are optimized once for all using the extreme values of the data set, before seeing the data. They thus can be ignored during the optimization of the histograms that exploit all the values of the data set.

Indexing terms. The new parameter i_{cen} (see Section A.3) is treated as a new indexing term with value fixed during the optimization of the central bin exponent (see Section A.3). Both the G-Enum-fp and G-Enum methods encode the number of intervals K using a $\log^* K$ term. The G-Enum method encodes the granularity G using the $\log^* G$ term, then exploits it in using only power of 2 granularities. With the G-Enum-fp, we suggest to encode the depth d of the hierarchy of granulated bins using a $\log^*(d+1)$ term. For a given depth d, the number of granulated bins that cover the data set is G_d, with $2^{d-1} < G_d \leqslant 2^d$ (see Section 4.2). We thus use G_d rather than $G = 2^d$ to be more parsimonious in the indexing terms. Note that this encoding that finely exploits the floating-point modeling space is actually very close from that of the G-Enum method, since at most one bit per interval can be saved.

Multinomial terms. Both methods need to encode the partition of the elementary bins into intervals and distribution of the n instances on the K intervals, and they rely on the same terms.

Bin index terms. With the G-Enum method, the *bin index terms* are used to encode the position of the h_k instances of each interval on the elementary ϵ-bins of the interval. As interval k consists of G_k granulated equal-width bins, each containing $\frac{E}{G}$ ϵ-bins, we obtain $E_k = G_k \frac{E}{G}$ ϵ-bins per interval, which translates into bin indexing terms $\sum_{k=1}^{K} h_k \log E_k = \sum_{k=1}^{K} h_k \log G_k + n \log \frac{E}{G}$. With the G-Enum-fp method, the length of each interval is also chosen using G_k granulated bins, except than these bins are floating-point bins and their size vary with the corresponding main bins. However, the length $L_k = c_k - c_{k-1}$ of each interval can also be expressed as $L_k = E_k \times \epsilon$, but we need to define ϵ precisely. Let ϵ be the minimum length over all the elementary floating-point bins, that is the length of the mantissa bins inside the main bin with the smallest exponent. All elementary bins have a length that a multiple of ϵ by a power of two factor. Since each histogram interval consists of a subset of elementary bins, the length L_k of each interval is a multiple E_k of ϵ, as in the G-Enum method.

Optimization. Overall, the G-Enum-fp criterion given in Tables 4, 5 is very close to the G-Enum criterion (see Table 1). Once the hyper-parameters terms and the central bin exponent are set, they can be ignored during optimization and the G-Enum-fp criterion becomes almost identical to the G-Enum criterion. This allows to inherit from its theoretical properties and to reuse its efficient optimization

Table 4
G-Enum-fp hyper-parameters criterion

Term	Criterion
Main bin containing the domain lower bound	$3\log 2 + \log^*(1+\lvert i_{lb}\rvert)$
Main bin containing the domain upper bound	$3\log 2 + \log^*(1+\lvert i_{ub}\rvert)$
Central bin exponent if necessary	$\mathbb{1}_{\{D_{lb}\leqslant 0<D_{ub}\}}(\log 2 + \log^*(1+\lvert i_{cb}\rvert))$
Digit precision used for mantissa bins	$\log^*(1+d_b)$
Mantissa bins containing the domain bounds	$2d_b \log 2$

Table 5
G-Enum-fp criterion

Criterion	Indexing terms	Multinomial terms	Bin index terms
G-Enum-fp	$\log^*(1 + i_{cen_{max}} - i_{cen}) + \log^* K + \log^*(1+d) + \log \binom{G_d+K-1}{K-1}$	$\log \binom{n+K-1}{K-1} + \log \frac{n!}{h_1!...h_K!}$	$\sum_{k=1}^{K} h_k \log E_k$

algorithm that mainly consists of a loop on granularities, and for each granularity a bottom-up greedy heuristic to optimize the intervals bounds. The overall algorithmic complexity is still $O(n \log n)$, since at most two values of the central bin exponent are tested (see Section A.3).

Note that the optimized histograms may contain extreme empty intervals if the optimal digit precision d is greater than the precision d_b used to encode the domain bounds. This means that the domain bounds can be refined if necessary after all values of the data set have been processed. In this case, we eliminate these extreme empty intervals in a post-processing step, as they do not provide useful information for exploratory analysis.

Appendix B. Properties of G-Enum-fp method

In this section, we further analyze the G-Enum-fp criterion and investigate on some of its properties.

B.1. Alternative formulations of the criterion

Let us focus on the bin index terms C_{bin} of the criterion of the G-Enum-fp criterion (see Table 5).

$$C_{bin} = \sum_{k=1}^{K} h_k \log E_k. \tag{20}$$

Let us recall that the interval boundaries $c_k, 0 \leqslant c_k \leqslant K$, are borrowed from a set of floating-point bins and that their length $L_k = c_k - c_{k-1}$ are multiples of ϵ. We have

$$\begin{aligned} C_{bin} &= \sum_{k=1}^{K} h_k \log \frac{L_k}{\epsilon}, \\ &= \sum_{k=1}^{K} h_k \log \frac{c_k - c_{k-1}}{\epsilon}, \end{aligned}$$

$$= \sum_{k=1}^{K} h_k \log (c_k - c_{k-1}) - n \log \epsilon. \quad (21)$$

Note that the length of the intervals can be potentially very large, up to DBL_MAX, and ϵ can be very small, down to DBL_MIN. Contrarily to the G-Enum method, the numbers $E_k = L_k/\epsilon$ cannot be directly used without raising numerical problems such as overflow. Formula 21 shows how to effectively compute the bin index term, provided that we keep track of the interval boundaries. And in the optimization algorithm, the constant term $-n \log \epsilon$ can be ignored.

Let us normalize the lengths of the intervals on the $[0, 1]$. Using this relative lengths instead of the absolute lengths, we get

$$C_{bin} = \sum_{k=1}^{K} h_k \log \frac{c_k - c_{k-1}}{D_{ub} - D_{lb}} + n \log \frac{D_{ub} - D_{lb}}{\epsilon}. \quad (22)$$

Interestingly, Formula 22 shows that the bin index term can be decomposed into a term that depends only of the relative lengths of the intervals, and a constant term that depends on the range $[D_{lb}, D_{ub}]$ of the value domain and of the precision ϵ and can be ignored during the optimization.

B.2. Advanced comparison with the G-Enum method

Hyper-parameters. While the G-Enum method needs only two extreme values x_{min} and x_{max} of the data set to define the elementary bins, the G-Enum-fp method requires two domain bounds D_{lb} and D_{ub} plus an additional parameter, i_{cb}, which is the exponent of the largest possible central bin that does not contain any non-zero value of the data set. The hyper-parameters were explicitly introduced in the G-Enum-fp method mainly to clarify some modeling choices and to avoid potential undesirable density side-effects around the domain bounds. Since the value of the hyper-parameters is constant whatever the model, the impact on the optimized histograms is likely to be negligible when the average quality of the estimated density is the only concern. Nevertheless, in a context of exploratory analysis, a better treatment of the density around the extreme values may be beneficial.

Central bin exponent parameter. Beyond the different representation spaces, floating-point bins versus equal-with bins, the G-Enum-fp method exploits the new parameter i_{cen} to choose the central bin exponent. This allows to consider a list of intermediate representation spaces, ranging from maximally floating-point, which may be suitable in the case of heavy-tailed distributions, to maximally equal-width which looks natural for distributions with bounded and not too large support.

The precision parameter. In the G-Enum method, the ϵ parameter is obtained simply by $\epsilon = (x_{max} - x_{min})/E$, where $E = G_{max} = 2^{30}$. In the G-Enum-fp method, the ϵ parameter is given by the length of the smallest mantissa bin with the main bin having the smallest exponent, obtained at the maximum depth $d_{max} = 30$.

Applicability to a huge range of values. The total number of ϵ-bin length intervals considered in the G-Enum-fp method is

$$E = \frac{D_{ub} - D_{lb}}{\epsilon}.$$

In the extreme case of a data set with $x_{min} = -$DBL_MAX, $x_{max} =$ DBL_MAX, $i_\star = -1022$ (as DBL_MIN $= 10^{-1022}$), the number of main bins used to cover the data set is maximum: $n_B = 2046 * 2 + 2 = 4094$.

As 12 bits are necessary to encode the index of the main bins, the mantissa bins can be encoded using up to 18 bits at the maximum depth $d_{max} = 30$, keeping a relative precision of around four millionths.

$$E \leqslant \frac{2 \times \text{DBL\_MAX}}{\text{DBL\_MIN}/2^{18}} = 2^{2064}$$

Compared to the G-Enum method that is limited to at most $2^{30} \approx 10^9$ ϵ-bin intervals, the G-Enum-fp method extends this limit by more than six hundred orders of magnitude, to $2^{2064} \approx 10^{621}$.

Boundary of intervals. Another difference is that all elementary or granulated bins have the same width in the G-Enum method, which means that the index of an elementary or granulated bin is sufficient to derive its bounds and its width. With the G-Enum-fp method, both the elementary and granulated bins have variable lengths, and we must keep track of the boundaries of each interval at the precision given by the current depth d to correctly compute the terms of criterion (see Formula 21). Finally, let us note that with the G-Enum method, the bounds of the elementary bins are calculated using a formula ($c_k = x_{min} - \epsilon/2 + k\epsilon$). Although this formula is very simple, it can lead to tricky numerical problems due to rounding problems on computers. With the G-Enum-fp method, the elementary bins are selected from the predefined set of floating-point bins. They are more complex to index easily, but their bounds are exact on computers and they does not suffer from rounding problems.

B.3. Invariance to linear transformation of the data.

When the data are linearly transformed, the resulting histogram model is expected to consist of the same intervals, with their boundaries linearly transformed in the same way. Formula 22 shows that optimized histograms depend only of the relative lengths of each interval, regardless of the range of the values. When the boundaries of the intervals are linearly transformed, their relative lengths remain unchanged and the optimization algorithm should result in the expected histogram.

However, there are differences that depend on the range $[x_{min}, x_{max}]$ of the data set, which can impact the optimization results. The definition of the elementary floating-point bins may involve a single main bin if the data is far from 0, or many main bins if data is close to 0 and require a central bin with a small exponent. The number of main bins n_B may therefore vary depending on the transformation of the data set, and the remaining bits $m = d_{max} - \lceil \log_2(n_B) \rceil$ used to encode the mantissa may change accordingly. This has an impact on the bin index term of the criterion which depends slightly on the accuracy of the interval boundaries. Another impact is that the prior term $\log \binom{G_d+K-1}{K-1}$ is likely to vary a lot when a data set far from 0 is translated to the origin. As $\log \binom{G_d+K-1}{K-1} \approx (K-1) \log G_d \approx (K-1) d \log 2$, this implies that a few more bits are needed per interval if a greater depth d is required to encode the interval boundaries with better accuracy. Note that these two impacts should be of secondary importance compared to the other terms of the G-Enum-fp criterion.

Overall, we can expect the histograms to be roughly invariant to the linear transformation of the data in a larger number of cases. Nevertheless, this property of invariance may be slightly violated in the case of data sets with values close to the origin and few instances.

B.4. Normalized criterion

In order to get a standardized criterion that evaluates the quality of a histogram, we suggest to normalize the cost a histogram model (the value of the corresponding evaluation criterion) by that of the null

histogram consisting of a single interval. We analyze the merits and limitations of the resulting criterion, and suggest some corrections to improve it.

The level criterion. Let $M_\emptyset$ be the null model with one single interval.

The cost of the null model is

$$cost(M_\emptyset) = cost_{hp}(D) + 3\log^*1 + n\log\frac{D_{ub} - D_{lb}}{\epsilon},$$

where $cost_{hp}(D)$ is the value of the criterion for the hyper-parameters, that is constant for a given data set D.

The standard *level* criterion is defined as

$$\text{level}(M) = 1 - \frac{cost(M)}{cost(M_\emptyset)}.$$

It can be interpreted as a compression ratio with values between 0 and 1. Its value is 0 when the best histogram consists in one single interval, which corresponds to a uniform density. And its value is close to 1 in the case of densities that are far from the uniform density.

Since the data range $(x_{max} - x_{min}) \leqslant (D_{ub} - D_{lb})$ can be arbitrarily large and ϵ arbitrarily small, the null cost can be arbitrarily large. As for the cost of a model, it can be very close to zero in the case of singular histograms, where most instances are in singular intervals of length ϵ. In the case of extreme outliers where the range of data is much larger than the range of the underlying distribution, the level criterion is likely to be close to 1.

Limits of the level criterion. Since histograms are approximately invariant to linear transformation (see Section B.3), one would expect the level criterion to be nearly constant in this case. Unfortunately, while the numerator $cost(M_\emptyset) - cost(M)$ should be approximately constant (because the term $n\log(D_{ub} - D_{lb})/\epsilon$ is common to both costs: see Formula 22), the denominator $cost(M_\emptyset)$ is subject to arbitrarily large variations. This results in a level criterion that can vary considerably in cases where its is supposed to be nearly constant.

Level criterion for floating-point histogram models. We suggest to adapt the level criterion by replacing its denominator by an invariant null cost term $cost_{D_{ref}}(M_\emptyset)$, related to a reference data set D_{ref} within the value domain $[1, 2]$ and containing the same number of instances. As D_{ref} is contained in one single exponent bin of length $2^0 = 1$, the maximum depth $d_{max} = 30$ corresponds to mantissa bins of length $\epsilon = 2^{-30} \approx 10^{-9}$. D_{ref} is therefore a data set where all elementary floating-point bins are of the same length, and it behaves identically to the G-Enum method that relies on equal-width elementary bins. We have

$$\text{level-fp}(M) = \frac{cost(M_\emptyset) - cost(M)}{cost_{D_{ref}}(M_\emptyset)}$$

with

$$\begin{aligned} cost_{D_{ref}}(M_\emptyset) &= cost_{hp}(D_{ref}) + 3\log^*1 + d_{max}n\log 2, \\ &= ((6 + 2d_b)\log 2 + \log^*1 + \log^*2 + \log^*(1 + d_b)) + 3\log^*1 + d_{max}n\log 2, \\ &= 4\log^*1 + \log^*2 + \log^*(1 + d_b) + (30n + 2d_b + 6)\log 2. \end{aligned}$$

For a given data set D, let us consider the linear transformation of D to D_{ref}. As the resulting histograms are approximately invariant to linear transformation, we have

$$\begin{aligned}\text{level-fp} &= \frac{cost_D(M_\emptyset) - cost_D(M)}{cost_{D_{ref}}(M_\emptyset)}, \\ &\approx \frac{cost_{D_{ref}}(M_\emptyset) - cost_{D_{ref}}(M)}{cost_{D_{ref}}(M_\emptyset)},\end{aligned}$$

which gives a stable normalized criterion. For most data sets, a relative accuracy of one billionth should be more than sufficient to reliably encode the relative lengths of the intervals. However, singular histograms may have singular intervals close to the origin, with a relative length less than 10^{-9}, resulting in a level value greater that 1 in extreme cases.

Overall, the level-fp criterion is a good candidate for assessing the quality of regular histograms for most data sets. This criterion may be greater than 1 in the case of pathological data sets involving singular intervals or outliers that are more than a billion times larger than the underlying distribution value range.

Appendix C. Experiments with common artificial data sets

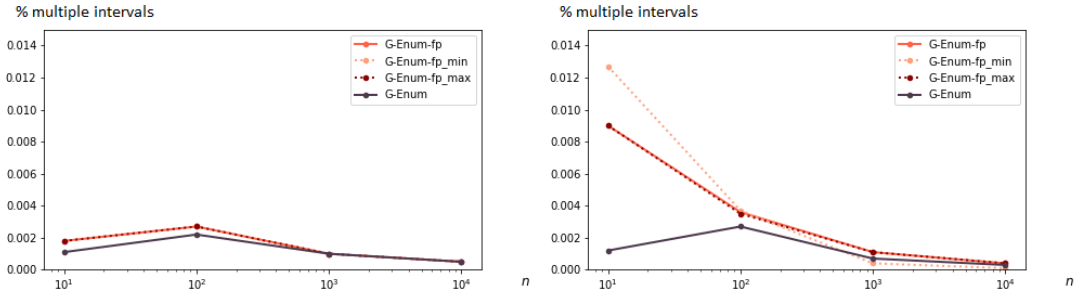
In this appendix, we evaluate the G-Enum-fp method in the case of common artificial data sets, using the uniform, normal and a mixture of normal distributions. The evaluation protocol is that of Section 6.1.

C.1. Uniform distribution

The objective of this experiment is to evaluate the robustness of the histogram methods, that should build one single interval in the case of a uniform distribution. A second objective is to finely compare the impact of the prior terms in each method criterion, and of the representation space, floating-point bins versus equal-width bins. We then exploit data sets of very small size as a magnifying glass to highlight the differences that are likely to disappear as the size of the data increases.

We generate data sets of size 10, 100, 1000, 10,000, using the uniform distribution $\mathcal{U}(1,2)$ and $\mathcal{U}(0,1)$. The experiment is repeated 10,000 times and we show in Figure 28 the percentage of cases where each method constructs a histogram containing more than one interval. To get insights on the impact of the central bin exponent parameter i_{cen}, we also report the results of the G-Enum-fp method obtained with $i_{cen_{min}}$ and $i_{cen_{max}}$.

Comparing the G-Enum-fp and G-Enum methods, the main result is that both methods are very robust, as they estimate the density using one single interval in most cases, as expected for a piecewise constant density estimator applied to a uniform density. The percentage of histograms having more than one interval is below 0.5% for $n = 100$ and decreases as n increases, both for the G-Enum-fp and G-Enum methods that behave similarly. However, in the case of tiny data sets of size 10 for the $\mathcal{U}(0,1)$ distribution, the G-Enum-fp method produces histograms with two or even three intervals in about 1% of the cases, up to ten times more frequently than the G-Enum method. In fact, as the experiments is repeated 10,000 times, some generated data sets show non-uniform patterns, for example with one instance far from all the others, resulting in two intervals. Below we discuss why these weak patterns are more frequently detected using the G-Enum-fp method.



(a) $\mathcal{U}(1,2)$ (b) $\mathcal{U}(0,1)$

Figure 28. % histograms with multiple intervals.

Uniform distribution $\mathcal{U}(1,2)$. With the $\mathcal{U}(1,2)$ distribution, both methods share the same representation space based on equal-width bins, as the $[1,2]$ value domain is covered by one single exponent bins with all mantissa bins of the same length. The only differences between the two methods are the prior terms that encode the granularity and the boundaries of the intervals:

$$\log^*(d+1) + \log \binom{G_d + K - 1}{K-1} \tag{23}$$

for G-Enum-fp versus

$$\log^* G + \log \binom{G + K - 1}{K-1} \tag{24}$$

for G-Enum. With one single exponent bin fully covered by the $[1,2]$ value domain, the number G_d of mantissa bins considered at each depth d is a power of two and we get $G_d = 2^d$ for G-Enum-fp, the same as the G term for G-Enum that considers only power of two granularities. Overall, the difference between the two methods reduces to $\log^*(d+1)$ for G-Enum-fp versus $\log^*(2^d)$ for G-Enum.

d	$G = G_d = 2^d$	$\log^*(d+1)$	$\log^* G$	$\log \binom{G_d+K-1}{K-1}_{K=2}$
0	1	1.05	1.05	0.69
1	2	1.75	1.75	1.10
5	32	4.11	7.17	3.50
10	1,024	5.27	12.04	6.93
20	1,048,576	6.43	20.20	13.86
30	1,073,741,824	7.11	27.85	20.79

Table 6

Prior terms for the granularity and the interval boundaries.

Table 6 gives the value of these prior terms for each method in the case of a histogram with $K = 2$ intervals. As a point of comparison, let us assume that a histogram is built using two intervals of lengths α and $(1-\alpha)$ with frequencies h_1, h_2. According to formula (22), the gain in coding length for the bin index terms is $h_1 \log \alpha + h_2 \log(1-\alpha)$ compared to the histogram with a single interval. With $n = 10$ and $\alpha = 1/2$, this amounts to $10 \log 2 \approx 6.93$, which is comparable to the values in Table 6.

Although the difference between $\log^*(d+1)$ and $\log^* G$ is small, it is not negligible in the case of tiny data sets. This slightly more parsimonious prior of the G-Enum-fp method explains why some weak patterns are slightly more frequently detected, as shown in Figure 28a.

Uniform distribution $\mathcal{U}(0,1)$. With the $\mathcal{U}(0,1)$ distribution, both methods exploit very different representation spaces, as the number of exponent bins involved in the G-Enum-fp method increases as more and more instances are generated near 0. Beyond the slight difference between their prior, the G-Enum-fp relies on floating-point bins of exponentially variable lengths, accurate around 0 and imprecise around 1. Encoding interval boundaries accurately is thus more expensive around 1 than around 0. This results in asymmetric behavior, depending on whether density patterns peak around 0 or 1, with more or less expensive trade-offs for more complex patterns. Figure 28b shows that the central bin exponent has a non negligible impact for the G-Enum-fp method in the case of tiny data sets, with multiple intervals about 50% more frequently for G-Enum-fp-min that exploits the maximally floating-point representation ($i_{cen} = i_{cen_{min}}$). Compared to the $\mathcal{U}(1,2)$ distribution that does not include the value 0, the G-enum-fp methods builds multiple intervals up to 10 times more frequently for tiny data sets. To put this difference in behavior into perspective, it should be remembered that this false detection of multiple intervals occurs in less than 1% of cases within the limit of data sets of the order of ten instances.

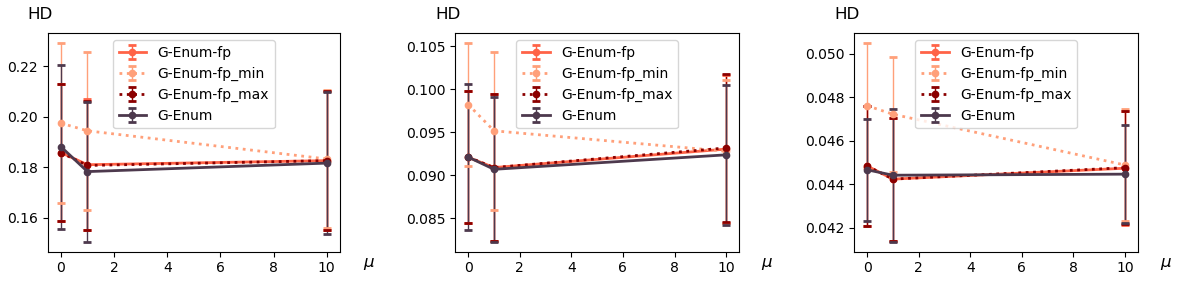
C.2. Normal distribution

The objective of this experiment is to empirically evaluate the invariance of histograms w.r.t. a linear transformation of the data, as analyzed in Section B.3. We generate data sets of size 100, 1000, 10,000 using the normal distributions centered around 0, 1 and 10. Whereas these data translations should have no impact for the G-Enum method, they involved important changes in the representation spaces for the G-Enum-fp method:

- $\mathcal{N}(0,1)$: many exponent bins are necessary to cover the data set symmetrically around the value 0,
- $\mathcal{N}(1,1)$: an asymmetric representation space is involved, as many exponent bins are necessary for values below 1 and very few for values above 1,
- $\mathcal{N}(10,1)$: most of the data fall in the exponent bin $]8,16]$, and the involved floating-point bins are mostly of the same length, as for the G-Enum method.

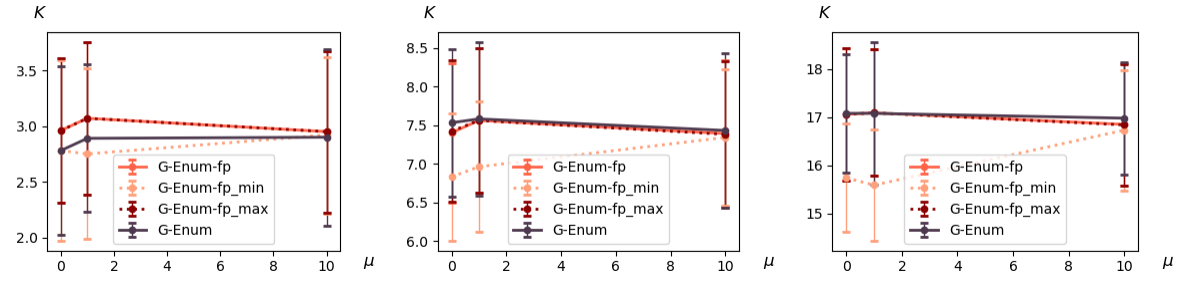
Beyond the G-Enum-fp method that optimizes the central bin exponent i_{cen}, we also analyze its variants obtained with $i_{cen_{min}}$ and $i_{cen_{max}}$. We compute the Hellinger distance between the densities estimated from the histograms and the underlying normal distributions. We also collect the number of intervals per histogram as an indirect measure of the quality of the histograms. Indeed, as the G-Enum-fp and G-Enum methods are regularized, they are not likely to overfit the data and the number of intervals appears to be highly correlated with the accuracy of the retrieved patterns. The experiment is repeated 100 times and we report in Figures 29,30 the mean and standard deviation of the Hellinger distance and number of intervals per sample size for each normal distribution $\mathcal{N}(\mu,1)$, with the mean μ on the X axis.

Both the G-Enum and G-Enum-fp methods give an increasingly accurate approximation of the normal distribution as the sample size increases, with an increasing number of intervals and decreasing Hellinger distance. As expected, the G-Enum method obtains results that are invariant to data translation, with almost the same mean number of intervals and Hellinger distance. As for the G-Enum-fp-min method, its maximally floating-point representation is stretched around 0, which is a singularity in the floating-point representation. This implies an increasing number of exponent bins for data sets that are dense around 0, and leads to more expensive prior terms to accurately encode interval boundaries throughout



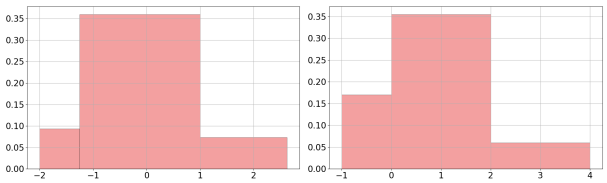
(a) $n = 100$ (b) $n = 1000$ (c) $n = 10,000$

Figure 29. Hellinger distance between histograms and normal distributions.



(a) $n = 100$ (b) $n = 1000$ (c) $n = 10,000$

Figure 30. Number of intervals per histogram for normal distributions.



(a) initial data set (b) data set translated by 1

Figure 31. Example of a data set from $\mathcal{N}(0,1), n = 100$, with an observable impact of translation.

the value domain. This is confirmed by Figures 29,30, which show that histograms contain about one less interval with a less accurate Hellinger distance as the mean of the normal distribution approaches 0. Note that these mean differences are not very large or significant given their standard deviation, but they are not negligible. Finally, the G-Enum-fp method optimizes the central bin exponent and almost always exploits the maximal one, corresponding to a maximally equal-width representation: its results are almost identical on average to those of the G-Enum method.

To exemplify a case where the difference can be observed, we choose the most unfavorable parameters according to Figures 29,30, with a tiny data set of size 100 generated from $\mathcal{N}(0,1)$ and its translation by 1. Figure 31 illustrates such a case, showing that the differences due to translation remain acceptable for exploratory data analysis.

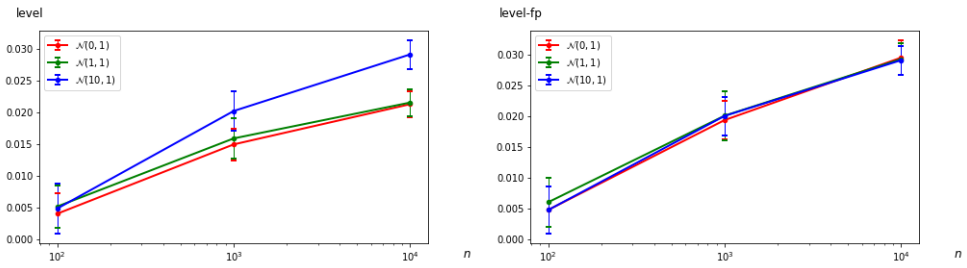


Figure 32. Evaluation criteria for histograms obtained with the G-Enum-fp method.

Finally, we display in Figure 32 the value of the criteria suggested in Section B.4 to evaluate the quality of histograms built using the G-Enum-fp method. The curves confirm that the *level* criterion is not stable and that its extended version *level-fp* is almost constant w.r.t data translation.

C.3. Normal mixture distribution

The objective of this experiment is to compare the histogram methods using a complex density distribution. We generate data sets of size $n = 10^i, 1 \leqslant i \leqslant 6$ using a mixture of ten normal distributions.

$$\sum_{i=1}^{10} \frac{1}{10} \mathcal{N}(10i - 5, 1) \tag{25}$$

The experiment is repeated 100 times and we collect the Hellinger distance and number of intervals per sample size.

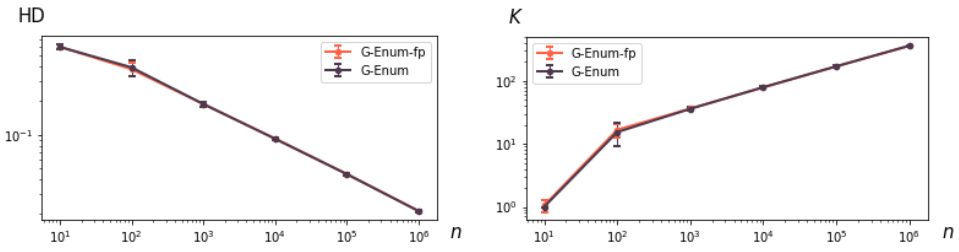


Figure 33. Hellinger distance and number of intervals for a mixture of ten normal distributions.

The overall results summarized using the mean and standard deviation of the Hellinger distance and number of intervals are shown in Figure 33. For all sample sizes, the two methods behave almost identically and the differences in representation space have an indistinguishable effect on the built histograms, as the the G-Enum-fp method almost always exploits the maximally equal-width representation, corresponding to the maximum central bin exponent.

Figure 34 shows an example of the histograms obtained by each method for $n = 10^6$.

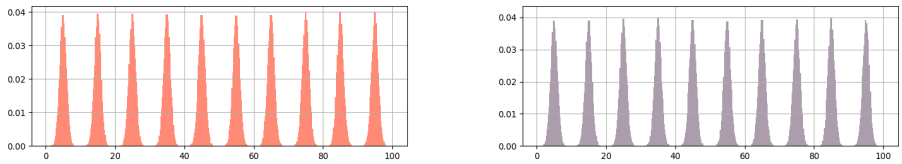


Figure 34. Histograms built with G-Enum-fp (left) and G-Enum (right) for a mixture of ten normal distributions.