Contemporary Tabular Data Generators Do Not Preserve Conceptual Structure

Anonymous authors
Paper under double-blind review

Abstract

Synthetic tabular data is usually judged by agreement with low-order marginals, by the loss of a downstream predictor, or by the failure of membership-inference attacks. Each is computed from a low-dimensional projection of the joint distribution, so none can tell whether a generator preserved the dependence structure among features. We measure this conceptual structure by treating a binary table as a formal context. We score 15 generators over multiple seeds, 13 learned and two structure-free baselines, on 63 real-world contexts. Each generated table is compared with its ground truth at matched density on the number of closed attribute sets, the size of the canonical implication base, and two bounded rise rates of the concept lattice. All 13 learned generators produce synthetic data with fewer closed sets and a smaller canonical base than their ground truth, and on these counts none is measurably better than a structure-free baseline. The two rise rates order the generators differently, and on them some learned generators are more faithful than one of the two baselines, so the second result holds in full for the counts only.

1 Introduction

Synthetic tables are used to share data that cannot be released, to train downstream models where real data is scarce, and to populate test databases and benchmarks, a use that predates current generative models (Gray et al., 1994; Li, 2020). The generators that produce them are now numerous, and they are judged against three targets: their output should agree with the real data on columns and low-order marginals, a classifier trained on it should transfer to the real data, and membership-inference attacks should fail.

All three targets are computed from a low-dimensional projection of the joint distribution (a marginal, a pairwise correlation or the loss of a single predictor). Two tables can agree on every column marginal and on the overall fill rate and still differ arbitrarily in which attributes occur together (Figure 1), and most of the uses above depend on which attributes occur together: a test database is queried through selections on combinations of attributes, and a downstream model learns from interactions between features. None of the three protocols reports whether those combinations survived generation. Rare rows are the most at risk,

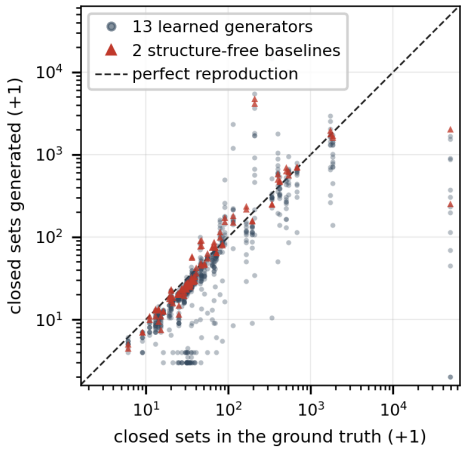

Figure 1: Closed attribute sets (sets of attributes that are exactly what some group of rows has in common) in each generated table against its ground truth, the real table it was fitted to, at the same density: 15 generators on 63 tables. Points below the diagonal have lost structure.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

since a pattern that occurs once barely moves
a marginal or a predictor's loss, so a generator can drop it at almost no cost on either target, yet it may be the only row that realises a whole group of attribute combinations. Several of the uses above are concerned with such rows (small subgroups in a released table, uncommon cases in training data and edge cases a test database has to exercise).

Recent evaluations document losses beyond the marginals. State-of-the-art generators do not fully preserve functional dependencies (Umesh et al., 2025) and frequently violate inter-column logical relationships (Long et al., 2025); work on high-order causal structure argues that evaluations built on low-order statistics leave multivariate relationships largely unmeasured (Senane et al., 2026); and the in-context conditionals of tabular foundation models need not cohere into any joint distribution (Klötergens et al., 2026). These studies test single dependency rules or low-order moments, so they do not show whether a generator loses individual rules or the combinatorial structure of the whole support (how many attribute sets are shared in full by some group of rows, and which implications hold between them).

We measure that structure with Formal Concept Analysis (FCA) (Ganter & Wille, 1999), which reads a binary table as a formal context and assigns it a concept lattice, its conceptual structure (Section 3.1). The lattice is determined by the positions of the filled cells, so its number of closed attribute sets and the size of its canonical implication base record information that no marginal contains. On contexts of the size used here, both counts can be enumerated exactly, so every generated table is compared with a known value. Distributional fidelity measures usually compare two samples through a classifier or a kernel statistic, and a small difference between generators then has to be separated from the variance of that estimate; with exact counts that step is not needed. Both counts also grow with density (Borchmann & Hanika, 2016), so each generated table is scored at the density of its source (the real context it was fitted to, which serves as ground truth). The measures apply to binary data only. The counts are isomorphism invariants, meaning a table can match them even if it does not contain any of the same rows as the original table. However, a copy of the source table will also match the counts, so they cannot be used instead of a privacy evaluation. Nevertheless, they do measure one aspect of the trade-off between covering the source's support and reproducing its records. We evaluate on 63 real binary contexts, and generators that emit continuous values are thresholded to the source density. Each of 15 generators draws 10 tables per context: 13 learned models and two baselines that model no dependence between attributes, one filling every cell at the source density and one also matching its row sums. Our contributions are an evaluation axis and two findings:

- C1. A structural evaluation axis. Generated tables are scored against their ground truth by exactly enumerated lattice counts, at matched density (Section 3).
- C2. Generators lose structure. All 13 learned generators undershoot the ground truth's closed sets and canonical base (median -0.30 and -0.75 bits), and the loss does not shrink as the ground-truth lattice grows (Section 4).
- C3. No learned generator beats a baseline on conceptual counts. Of 26 pre-specified comparisons with a structure-free baseline, 9 favour the baseline and 17 are equivalent within ± 0.5 bits. However, this can not be observed on the rise rates (Section 4.3).

Our claim concerns the evaluation protocols: the structure that they leave unmeasured is not preserved by any of the generators that we evaluate. Based on this evidence, the losses reported by the studies for single dependency rules cited above also apply to the combinatorial structure of the support over all attributes. However, the findings have a limited scope as they do not indicate that any particular generator is poor, nor do they constitute a ranking. We observe that resampling the corpus moves individual positions by up to 11 ranks, and that the rank of none of the 15 generators stays fixed (Section 4.2).

We consider the result relevant beyond the 13 models evaluated here, for two reasons. First, the loss is shared by generators that have little else in common: adversarial, variational, diffusion, tree-ensemble and autoregressive models, and in-context foundation models that are not trained on the target table at all. Its cause is therefore more likely to be something these models have in common, such as weighting row patterns by how often they occur, than a property of one architecture. Secondly, benchmarks influence the development of

future models: if a baseline that ignores co-occurrence is not outperformed in terms of structure, it suggests that the protocols reward something other than what their users want them to measure. Consequently, models tuned against these protocols will perpetuate this discrepancy. In addition, our two kinds of structural measure, the counts and the rise rates, rank the generators very differently, so structural fidelity cannot be summarised by a single number, and it remains open which of its aspects matters for a given use of synthetic data.

Several questions follow that this paper does not answer. The first is the cause: whether the loss comes from the training objective, from model capacity, or from how little data a single table offers, and whether it falls mostly on the rarest row patterns (Section 5). The second is the consequence: whether lost structure makes predictors trained on generated tables worse, especially on small subgroups, and whether it changes the rules mined from them. The third is scope: whether the result holds on categorical and mixed data and on tables too large for the lattice to be enumerated (Section 6). Further, it is open whether published rankings change once a structure-free baseline and a structural statistic are reported, and whether the loss in foundation models is related to the incoherence of their conditionals (Klötergens et al., 2026). Finally, C2 and C3 leave open whether or to what extent a generator can retain the structure of its source without copying it, or how much of the source's support it can cover before it reproduces records (Section 5).

2 Related Work

This paper brings together two areas of research: a rapidly evolving body of generative models for tabular data and a much slower-moving body of protocols for evaluating their effectiveness.

The problem of learning a sampler for a possibly mixed-type table of exchangeable rows has been approached with most of the major generative architectures. CTGAN and TVAE (Xu et al., 2019), distributed through the Synthetic Data Vault (Patki et al., 2016), established the modern setting; CTAB-GAN (Zhao et al., 2021) extended their encoding; diffusion followed (Kotelnikov et al., 2023; Mueller et al., 2025; Zhang et al., 2024; Shi et al., 2025). Autoregressive factorisations over discretised columns (Tiwald et al., 2025) and language-model serialisations of rows (Solatorio & Dupriez, 2023) form a third line, and tree ensembles a fourth, sampling from the leaves of an adversarially fitted forest (Watson et al., 2023) or from a diffusion whose score network is replaced by gradient-boosted trees (Jolicoeur-Martineau et al., 2024). Tabular foundation models (Hollmann et al., 2023; 2025; Grinsztajn et al., 2025; 2026; Ma et al., 2025; Qu et al., 2025) are a more recent line; they consume a small dataset in context and need no gradient step, but are trained as predictors, so they have no sampling procedure of their own: a table is produced column by column, each drawn from the model's in-context conditional given the columns already generated. Klötergens et al. (2026) find such conditionals need not cohere into any joint distribution. SMOTE (Chawla et al., 2002) predates all of this and models no distribution at all, interpolating between same-class nearest neighbours.

Three families of evaluation protocol recur across the above papers (Du & Li, 2025; Stoian et al., 2026). Statistical fidelity compares the real and synthetic marginals, typically with the Kolmogorov–Smirnov test or the total variation distance, and the association differences obtained from a correlation or mutual information matrix. Alaa et al. (2022) sharpen it to the sample level (α-precision, β-recall and authenticity), so that individual records can be audited as well as the distribution as a whole. Downstream utility is reported as the gap to a real-data-trained predictor, with training on synthetic data and testing on real data. Privacy is assessed by proxies for memorisation, most commonly the distance from each synthetic record to its closest real one, and membership-inference attack success. Toolkits such as Synthcity (Qian et al., 2023) and the Synthetic Data Vault (Patki et al., 2016) bundle these metrics, which largely explains why the protocol is so uniform across papers.

The question of whether methods for generating synthetic data from tabular data preserve the dependencies between attributes (beyond statistical correlations) has received far less attention. In classical machine learning, a few papers investigate logical and functional dependencies in the synthetic data directly: Umesh et al. (2025) introduce logical dependencies

alongside functional ones and find that state-of-the-art generators do not fully preserve the latter, while Long et al. (2025) propose three metrics for inter-column logical relationships that generators frequently fail to maintain, and Senane et al. (2026) argue that evaluations built on low-order statistics leave a model's higher-order causal structure largely unmeasured. A related direction feeds structural mismatch between real and synthetic data back into training (Chen et al., 2019; Umesh et al., 2026): the former designs a GAN whose generator and discriminator preserve the source table's functional dependencies, the latter reconstructs dependent features from predefined dependency rules.

A small body of literature studies randomly generated binary tables. The models studied there usually generate contexts with no dependence between cells (Borchmann & Hanika, 2016; Felde & Hanika, 2019). These papers report the distributions of intent and pseudo-intent counts. Analytically, Emilion & Lévy (2009) give the expected lattice size, Sakurai (2021) a superpolynomial lower bound on it, and Albano & Chornomaz (2017) an upper bound by the largest contranominal scale. Such generators also serve as test data for FCA algorithms (Rimsa et al., 2013), and swap randomisation (Gionis et al., 2007) serves as an analogous null model outside FCA.

We are not aware of prior work using lattice-level invariants to evaluate learned generative models, nor of any other established structural counterpart to the marginal, utility and privacy axes above.

3 Method

3.1 Theoretical Foundations

Every table in this paper is a binary matrix, and every generator we evaluate is asked to produce another one. Formal Concept Analysis is a method for investigating exactly such tables, and it supplies the tools we measure with below: it names the combinatorial structure of a binary matrix, which attribute sets some group of rows shares and which implications hold between them. No prior familiarity with the method is assumed, and readers interested in the mathematical background are advised to consult Ganter & Wille (1999).

A formal context is a triple $\mathbb{K} = (G, M, I)$ where G is a set of objects, M a set of attributes, and $I \subseteq G \times M$ an incidence relation; $(g, m) \in I$ is read "object g has attribute m". Concretely, $\mathbb{K}$ is a binary matrix with rows G and columns M. For $A \subseteq G$ and $B \subseteq M$ the two derivation operators are

$$A' = \{\, m \in M \mid (g, m) \in I \text{ for all } g \in A \,\}, \qquad B' = \{\, g \in G \mid (g, m) \in I \text{ for all } m \in B \,\}.$$

A formal concept is a pair (A, B) with $A' = B$ and $B' = A$; A is its extent and B its intent. Ordering concepts by extent inclusion yields the concept lattice $\underline{\mathfrak{B}}(\mathbb{K})$, the canonical structural summary of $\mathbb{K}$. For this paper, the relevant property is that the lattice depends on which cells are filled: two matrices can agree on every column marginal and on the overall fill rate and still have concept lattices of very different size and shape. The intents are exactly the closed attribute sets of the introduction, the sets B with $B = B''$, and their number equals the number of concepts.

A context is clarified when no two objects share an intent and no two attributes share an extent; clarification merges duplicate rows and columns, keeping one representative of each class. An object or attribute is reducible when removing it leaves the concept lattice unchanged up to isomorphism, and reduction removes all reducible rows and columns (Ganter & Wille, 1999). Both operations leave the lattice unchanged, so the clarified, reduced context is the canonical representative of $\underline{\mathfrak{B}}(\mathbb{K})$, and every lattice-level quantity is unaffected by which of its representatives is measured.

An implication $B_1 \to B_2$ over M holds in $\mathbb{K}$ if every object having all attributes in B_1 also has all attributes in B_2. The implications holding in $\mathbb{K}$ can be numerous, but all of them follow from a base, and the canonical base is the unique minimum-cardinality base and contains exactly one implication per pseudo-intent (Guigues & Duquenne, 1986), a set $P \subseteq M$ with $P \neq P''$ containing the closure of every pseudo-intent properly contained in it. Counting pseudo-intents and measuring the size of the base are therefore the same

measurement, and we report only the former. It is well known that enumerating pseudo-intents is computationally hard (Distel & Sertkaya, 2011), and both the number of intents and the number of pseudo-intents are bounded by $2^{|M|}$. The density of a formal context is its fill rate $|I|/(|G| \cdot |M|)$, with the degenerate case $|G| = |M| = 0$ assigned density zero. Both counts tend to grow with density: a denser matrix produces more closed sets whether or not the generator captured any structure, and in randomly generated contexts the counts largely track the density at which the contexts were produced (Borchmann & Hanika, 2016).

In a finite lattice an element with exactly one upper cover is meet-irreducible, one with exactly one lower cover join-irreducible; together, the irreducibles determine the lattice. Besides the two counts, we use the non-unit meet- and join-rise rates, lattice-shape statistics introduced as tools for investigating local distributivity (Abdulla et al., 2025). For two comparable lattice elements $x \leq y$, the meet-rise is the drop in the number of meet-irreducibles above them, $\Delta_m(x, y) = m(x) - m(y)$, and the join-rise the corresponding rise in join-irreducibles below them, $\Delta_j(x, y) = j(y) - j(x)$. On a clarified and reduced context, these equal the differences in intent and extent cardinality between the two concepts. A rise is unit when it equals one, and the two rates report the proportion of covering pairs with non-unit rises. A finite lattice is distributive if and only if every covering pair has unit meet- and join-rise, so the two rates describe how far, and in which direction, a lattice departs from distributivity. Whereas the counts describe the size of the lattice, the rates describe its shape. On real-world contexts, the rates show lattices that are far more join- than meet-distributive, with an asymmetry that grows with lattice size (Abdulla et al., 2025).

3.2 Experimental Pipeline

We use 63 real-world binary formal contexts drawn from two public collections: the FCA contexts repository (Hanika & Jäschke, 2024), a curated corpus of contexts used in the FCA literature, and KONECT (Kunegis, 2013), whose small networks we read as incidence matrices (bipartite) or adjacency matrices (unipartite). Each context is clarified and reduced before use (Section 3.1), so that no generator is credited for reproducing redundancy. We keep the corpus small (contexts from 3×3 to 68×62) so that the structural quantities can be computed exactly and each generated table is compared with its true value.

Not every method surveyed in Section 2 can be run on this corpus. Of the 20 considered generators for inclusion in the pipeline, 3 need a long per-dataset or per-corpus training run that we did not integrate and are not run. Two further methods return no output on this corpus: SMOTE interpolates between neighbours of the same class and has nothing to synthesise when class balancing asks for no new rows, and CDTD requires at least one continuous column, which a formal context by definition does not have. As both refusals are properties of the input, not failures of the implementation, we are left with 13 learned generators, listed with their implementations in Appendix B. Each is fitted on a single context, presented as a table of $|G|$ rows and $|M|$ binary columns, and asked for a synthetic table with as many rows as its source. Most generator libraries expose a supervised fit(X, y) interface, so the last attribute column is designated as the target; the choice is arbitrary, but it is the same column for every generator and so cannot favour one over another. We draw 10 independent tables per (context, generator) pair, with seeds derived deterministically from the triple (global seed, context name, table index), so that they do not depend on execution order (Appendix G).

To calibrate the learned generators, we fixed two structure-free baselines before any result was seen, Cointoss and Dirichlet (Felde & Hanika, 2019), both fitted to the source context. Cointoss reads only the density of the source and fills each cell independently with that probability. Dirichlet also fits the distribution of row sums, that is, of how many attributes each object has, and gives each object the sampled number of ones in uniformly random columns. Both match the source's marginals in expectation and destroy all co-occurrence structure, so neither can represent an implication, a non-trivial concept or any dependence between attributes. We use them as reference points: the margin by which a learned generator beats them measures what its training contributes beyond the density and row sums, which can be read off the source in a single pass.

All our measures are defined on a binary incidence relation, but several generators emit continuous values even when they are trained on 0/1 data, and their output has to be binarised before it can be measured. As mentioned earlier, both measured counts correlate with density. Therefore, we aim to make comparisons at matched densities. To achieve this, we apply a threshold to each table to make its density as similar as possible to that of its source, so that differences in the counts come from which cells are filled. This density matching step is applied to the 3 generators that emit continuous values. On every binary matrix, source or generated, we then compute the density, the intent count, the pseudo-intent count and the two non-unit rise rates, all with conexp-clj (Hanika & Hirth, 2019). Since the lattice can grow exponentially in the number of attributes and pseudo-intents are hard to enumerate, each measurement runs under a fixed time budget, 60 s per measure on a generated matrix and 1800 s on a source context; a measurement that exceeds it is recorded as missing, together with the reason.

The measures are on different scales, and we compare them differently. The two counts are on a ratio scale and span several orders of magnitude, so a generated count v is compared with the source count v_0 by the smoothed log-ratio $d_{\text{count}} = \log_2((v+1)/(v_0+1))$. The rise rates and the density are proportions in $[0, 1]$ and are compared by the difference $d_{\text{prop}} = v - v_0$ in their own units. On both scales, positive values mean that the generator overshoots the source and negative values that it undershoots it; a count deviation of -1 means roughly half as many closed sets as the source. A generator's structural-fidelity score is the median of $|d_{\text{count}}|$ over both count measures and all its tables, so lower is closer to the source and a score of 1 means off by a factor of two. The rise rates are pooled the same way into a second score, kept separate from the counts. Some generators need a minimum number of rows and do not run on the smallest contexts, so scores and ranks are computed on the 50 of the 63 contexts that every generator covers. These are also the contexts with structure to lose: on the smallest ones most generators reproduce the few closed sets exactly.

4 Results

The results follow the order of the contributions. Section 4.1 shows that every generator loses structure and that the learned generators lose the most (C2), Section 4.2 compares the learned generators with the two baselines (C3), and Section 4.3 reports the rise rates, on which C3 holds only in part. The figures show the deviation of the measures of every (context, generator) pair from the ground truth of its source. The largest median absolute density difference over all generators (whether binarised or not) is 0.096. Therefore, the differences reported below cannot be explained by density alone. We provide detailed reports in Appendix C and D.

4.1 Every generator loses structure, the learned ones more

All 13 learned generators produce fewer closed sets than their sources (Figure 2), with a median deviation of -0.30 bits across them. 13 of the 13 also produce a smaller canonical base, with a median of -0.75 bits. The largest losses are TABARGN's, at -3.00 bits on closed sets, and TVAE's, at -2.41 bits on pseudo-intents. The two baselines lose structure as well. Their tables have fewer closed sets than the source on most contexts, with medians of -0.09 bits for Cointoss and -0.14 bits for Dirichlet, and -0.25 and -0.14 bits on pseudo-intents. The sign of the deviation is therefore the same for learned generators and baselines, and they differ in its size: 13 of the 13 learned generators have a lower median than both baselines on closed sets, and 13 on pseudo-intents, where the gap is widest.

The loss does not shrink as lattices grow (Figure 3), and it is most extreme on the largest lattice in the corpus: its source (konect_league_be1_2016) has 49 152 closed sets, the median generator reproduces 254 of them, and TABARGN returns a matrix whose lattice has a single concept. Grouped by the number of closed sets in the source, the learned generators' median closed-set deviation is -0.46 bits below 30, -0.33 bits between 30 and 1000, and -1.36 bits above 1000. The baselines' medians in the same groups are -0.24, -0.11 and -0.15 bits, so the learned generators lose more in every group. The top group holds

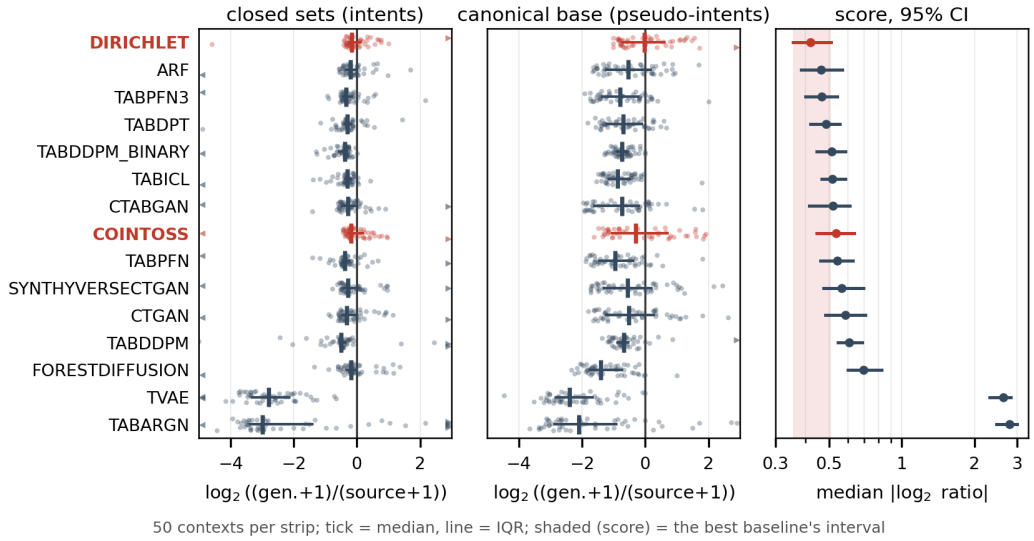

Figure 2: How far each generator's tables deviate from their sources on the two counts, at the source density. Strips: one dot per (context, generator), the median over its seeds of $\log_2((v+1)/(v_0+1))$; dots left of zero are contexts where the generator lost structure. The tick is the generator's median, the line its interquartile range. Right: the structural-fidelity score with its 95% cluster-bootstrap interval over contexts (log axis); the red band is the best-scoring baseline's interval. Rows are ordered by the score.

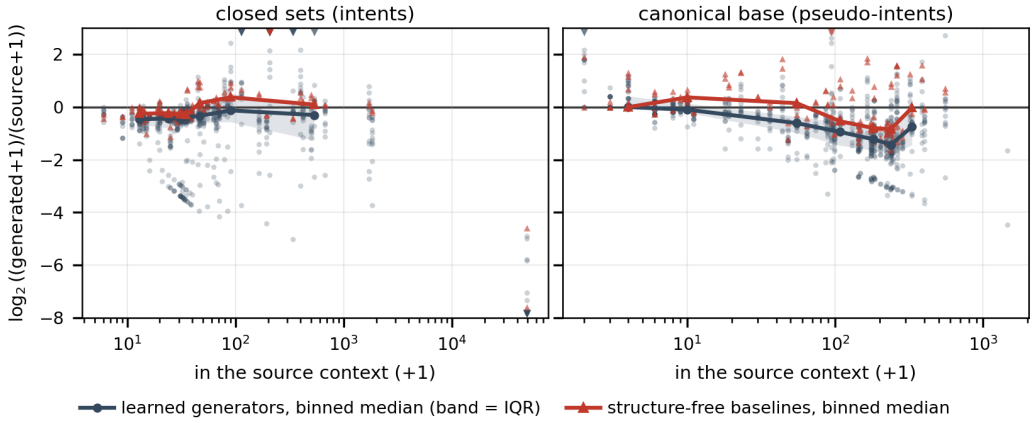

Figure 3: Deviation from the source against the size of the source lattice. One point per (context, generator) cell on every context the generator covers; cells beyond the axis are drawn as triangles on its edge. Lines are medians within equal-count bins of at least eight contexts, for the learned generators (band: interquartile range) and for the two baselines.

only 3 contexts, too few to establish a trend, but they show that the result is not produced by the small contexts that make up most of the corpus.

4.2 No learned generator beats a structure-free baseline

To test C3, we compared each baseline with each learned generator on the contexts both cover, which gives 26 comparisons, fixed before any score was inspected. Each has a paired cluster-bootstrap interval for the difference in score and an exact sign test over contexts, Holm-corrected over the whole family (Table 9). After correction, 9 comparisons favour the

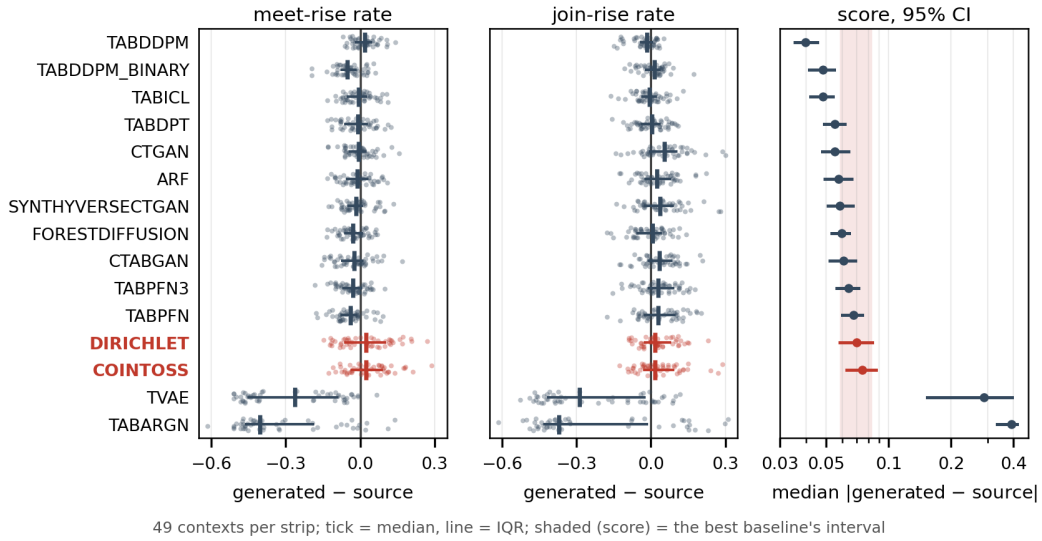

Figure 4: Figure 2 for the two rise rates. Strips: each (context, generator) cell's difference from the source rate (generated minus source, median over seeds). Right: the rise-rate score, the median absolute difference over both rates, with its 95% cluster-bootstrap interval (log axis). Rows are ordered by this score, not by the count score. Figure 6 (Appendix F) shows non-unit rises on a concrete lattice.

baseline and 0 favour the learned generator. The largest difference is between DIRICHLET and TVAE: 1.85 bits ([1.49, 2.25]), with the baseline closer to the source on 57 of 60 contexts and $p < 0.001$ after correction. In the remaining 17 comparisons the whole interval lies within ± 0.5 bits. This bounds the gap, though loosely, since 0.5 bits is a factor of $2^{0.5}$ in the count. This does not mean that the baselines model anything, since they cannot represent any dependence between attributes. It means that, on the counts, whatever the learned generators capture beyond density and row sums does not bring their tables closer to the source than the baselines' tables. Among the generators themselves, the ordering is mostly unresolved: the score intervals in Figure 2 overlap for most pairs, and recomputing every generator's rank within each of 10000 bootstrap replicates over contexts gives rank intervals up to 11 positions wide, and the resampling fixes the rank of none of the 15 (Table 8). Dirichlet has the lowest score, with a rank interval of [1, 6]; Cointoss has [2, 12]. We therefore make no claim about the position of any single generator, and the order of the rows in Figure 2 is not to be read as a ranking.

4.3 The rise rates order the generators differently

The rise rates describe the shape of the lattice, and they order the generators differently from the counts (Figure 4; rank correlation 0.24). Run on the rise-rate score, the same 26 comparisons decide 4 for a baseline and 4 for a learned generator after correction, where the counts decide 9 and 0. The two scores never disagree on a pair that they both decide. The 4 comparisons that favour a baseline (TABARGN and TVAE, against both baselines) are also decided for the baseline on the counts. The 4 that favour a learned generator (ARF, TABDDPM, TABDDPM_BINARY and TABICL) are all against Cointoss, and the counts find each of them equivalent (Appendix E). Thus we can not state with significance that Dirichlet beats the learned generators when considering rise-rates. Apart from TABARGN, whose score is 0.391, the rise-rate scores of all generators lie between 0.040 and 0.288, and the two baselines, ranked 1 and 8 on the counts, rank 12 and 13 of 15 here. The rise-rate ordering is stable across context subsets (Spearman 0.94) but undefined on konect_league_be1_2016, the largest lattice, whose rise rates exceeded the measurement budget. Figure 5 (a close parallel to Figure 3) is provided in Appendix E, in which the rise-rate deviations are plotted against lattice size.

5 Discussion

The learned generators lose more structure than the baselines, although they are trained on the source table while the baselines read only its density and row sums. Some loss is expected of any table filled at random at the source density, since it does not reproduce the correlations between attributes. For the additional loss we offer an explanation consistent with C2 and C3, which remains a hypothesis because the benchmark observes only the generated tables. The training objectives of the learned generators weight a row pattern by how often it occurs. In a 40-row context, a pattern that occurs once contributes a fortieth of the likelihood and costs little to drop, yet dropping it can remove a whole branch of the lattice with the implications it supports. A structure-free baseline does not favour frequent patterns, so it has no reason to drop rare ones. The hypothesis predicts that the lost closed sets are those supported by the fewest rows, which can be tested on the released data (the first open question of Section 1). Two observations speak against simpler explanations. The loss appears in every model family, including foundation models that are not trained on the target table, so it is not a property of one architecture; and it does not depend on how binary columns are presented: the two TabDDPM variants differ only in whether the columns are declared discrete, and their median closed-set deviations are -0.49 and -0.39 bits.

For evaluation, a metric on which a baseline calibrated to the density is competitive does not measure what its readers take it to measure. Both of our baselines match the source density by construction, so an evaluation that stops at marginals rates them like a diffusion model, although each takes a few lines of code. We therefore recommend reporting such a baseline in every evaluation and, for binary and categorical data, at least one lattice statistic, such as the number of closed sets, always at matched density, since otherwise the statistic becomes a second measure of density.

For the generators themselves, two untested directions follow: a training term that rewards support coverage, charging the model for attribute combinations that occur in the source but not in its output however rare they are, or generating the structure first, by fitting an implication base and sampling rows consistent with it. Both push a generator towards reproducing rare rows, which are also the easiest to re-identify, so either has to be evaluated together with privacy.

6 Limitations

The corpus is small because exact enumeration requires it: contexts range from 3×3 to 68×62. We therefore cannot show the loss at the sizes at which synthetic data is typically used, where both counts are intractable. Within the corpus the loss does not shrink with lattice size, but the group of largest lattices holds only 3 contexts, and 33% of the corpus is one family of similar wiki-edit networks (Appendix A). Since every generator is scored on the same contexts, this cannot favour one of them, but it limits how far the effect sizes generalise. A formal context is binary, whereas most tabular benchmarks, and several of the generators, are mixed-type; a structural evaluation of such data needs a different measure, for example conceptual scaling of a many-valued context, which we do not test. Finally, we ran no privacy evaluation.

Every generator runs at its library defaults, apart from the few settings in Appendix B. Tuning can only help the learned generators, since the baselines have no hyperparameters, and could in principle change C3, but the usual protocols offer no structural target to tune towards, and the marginal target they do offer is already met at the defaults (Section 3). C3 holds in full for the counts only: on the rise rates 4 learned generators beat Cointoss, and the baselines rank 12 and 13 of 15 (Section 4.3); the rise rates, however, are undefined on the largest lattice and cannot replace the counts as a ranking.

Reproducibility statement

We release the full pipeline (context preparation, sampling, binarisation, measurement, aggregation, tables and figures) as code. Sampling is controlled by a single --seed argument: the seed of each sample is derived from the global seed, the context name and the sample index, so it does not depend on the processing order. A test verifies that every generator evaluated here is bit-reproducible under this seed, and all dependencies are pinned by a lock file. Since enumerating pseudo-intents is hard, measurements run under a fixed time budget, and the released cache records every measurement that exceeds it together with the reason and the budget. Appendix G gives the details. Appendix C accounts for every measurement that is expected but absent. The generators, binarisations, measures, deviation scales and the matched-density rule are specified in Section 3 and Appendix B, and the resampling scheme, the pre-specified comparison family and the multiplicity correction in Section 4.2. The code and every artifact needed to regenerate the tables and the numbers in the text (the prepared source contexts and the aggregated measurement cache, which also covers the contexts of Appendix A) are submitted as supplementary material.

AI use statement

This work was carried out with substantial help from a general-purpose LLM-based coding agent. We describe its use in the categories of the ICLR AI policy.

Required-disclosure categories. The agent served as a discussion partner while we designed the evaluation protocol, in particular for the decisions stated in Section 3: separate deviation scales for bounded proportions and ratio-scale counts, and fixing the family of baseline comparisons before any score was inspected. Each decision is argued in the text and can be judged there. We did not use generative AI to produce any data analysed in this paper: every synthetic table was produced by one of the generators of Section 3, running locally, and no LLM generated, filtered or edited a table, a measurement or a number quoted in the text. The paper contains no theorems, and all statistical claims are computed by the released code from the released cache.

Recommended-disclosure categories. Under our supervision, the agent wrote a large part of the released implementation (the sampling, binarisation, measurement and aggregation stages, the statistical module and the table and figure generators) and substantial parts of the prose, which we then revised where necessary. It was also employed for the purpose of searching and summarising publications for a preselection of related work for Section 2.

Checks. Errors in AI-written analysis code are hard to see, because such code still produces plausible numbers. We therefore built the artifact so that such errors become visible. Our aim was that no number in the text would be typed by hand. Each number is a macro emitted by the code that builds the tables from the same measurement database. This means that the text and tables cannot disagree. Where a sentence states a property of the data (for example, that every learned generator undershoots its source), the code re-checks this property on a full run and warns if it no longer holds. The full pipeline, including the reproducibility test of Appendix G, is released, so every reported quantity can be recomputed. The authors reviewed the implementation and the manuscript, and checked the cited works against their primary sources.

Responsibility. We take full responsibility for the content of this work, including text, claims and artifacts produced with the help of generative AI.

References

Mohammad Abdulla, Tobias Hille, Dominik Dürrschnabel, and Gerd Stumme. Rises for measuring local distributivity in lattices. In Conceptual Knowledge Structures (CONCEPTS 2025), Lecture Notes in Computer Science. Springer, 2025.

Ahmed M. Alaa, Boris van Breugel, Evgeny S. Saveliev, and Mihaela van der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In Proceedings of the 39th International Conference on Machine Learning (ICML), volume 162 of PMLR, 2022.

Alexandre Albano and Bogdan Chornomaz. Why concept lattices are large: Extremal theory for generators, concepts, and VC-dimension. International Journal of General Systems, 46(5):440–457, 2017.

Daniel Borchmann and Tom Hanika. Some experimental results on randomly generating formal contexts. In Proceedings of the Thirteenth International Conference on Concept Lattices and Their Applications (CLA 2016), volume 1624 of CEUR Workshop Proceedings, pp. 57–69. CEUR-WS.org, 2016.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16:321–357, 2002.

Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. Faketables: Using gans to generate functional dependency preserving tables with bounded real data. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 2074–2080. International Joint Conferences on Artificial Intelligence Organization, 2019.

Felix Distel and Barış Sertkaya. On the complexity of enumerating pseudo-intents. Discrete Applied Mathematics, 159(6):450–466, 2011.

Yuntao Du and Ninghui Li. Systematic assessment of tabular data synthesis. In Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS '25, pp. 2414–2428, New York, NY, USA, 2025. Association for Computing Machinery.

Richard Emilion and Gérard Lévy. Size of random Galois lattices and number of closed frequent itemsets. Discrete Applied Mathematics, 157(13):2945–2957, 2009.

FCA Tools Working Group. contexts: A collection of formal contexts (datasets). https://github.com/fcatools/contexts, 2024. Also available at https://fcarepository.org/.

Maximilian Felde and Tom Hanika. Formal context generation using dirichlet distributions. In Dominik Endres, Mehwish Alam, and Diana Şotropa (eds.), Graph-Based Representation and Reasoning, Lecture Notes in Computer Science, pp. 57–71. Springer, 2019.

Bernhard Ganter and Rudolf Wille. Formal Concept Analysis: Mathematical Foundations. Springer, Berlin, Heidelberg, 1999.

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

Jim Gray, Prakash Sundaresan, Susanne Englert, Kenneth Baclawski, and Peter J. Weinberger. Quickly generating billion-record synthetic databases. In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, pp. 243–252, 1994.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, et al. TabPFN-2.5: Advancing the state of the art in tabular foundation models. arXiv preprint arXiv:2511.08667, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, et al. TabPFN-3: Technical report. arXiv preprint arXiv:2605.13986, 2026.

Jean-Louis Guigues and Vincent Duquenne. Familles minimales d'implications informatives résultant d'un tableau de données binaires. Mathématiques et Sciences Humaines, 95: 5–18, 1986.

Tom Hanika and Johannes Hirth. Conexp-clj – a research tool for FCA. In Supplementary Proceedings of ICFCA 2019 Conference and Workshops, volume 2378 of CEUR Workshop Proceedings, pp. 70–75. CEUR-WS.org, 2019.

Tom Hanika and Robert Jäschke. A repository for formal contexts. In Conceptual Knowledge Structures (CONCEPTS 2024), Lecture Notes in Computer Science. Springer, 2024.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In International Conference on Learning Representations (ICLR), 2023.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. Nature, 637(8045):319–326, 2025.

Alexia Jolicoeur-Martineau, Kilian Fatras, and Tal Kachman. Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees. In Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS), volume 238 of PMLR, pp. 1288–1296, 2024.

Christian Klötergens, Vijaya Krishna Yalavarthi, Lars Schmidt-Thieme, and Tom Hanika. Do tabular foundation models agree with themselves? arXiv preprint arXiv:2608.06004, 2026.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In Proceedings of the 40th International Conference on Machine Learning (ICML), volume 202 of PMLR, 2023.

Jérôme Kunegis. KONECT: the Koblenz network collection. In Proceedings of the 22nd International Conference on World Wide Web (WWW '13 Companion), pp. 1343–1350. ACM, 2013.

Wanxin Li. Supporting database constraints in synthetic data generation based on generative adversarial networks. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, pp. 2875–2877, 2020.

Yunbo Long, Liming Xu, and Alexandra Brintrup. Evaluating inter-column logical relationships in synthetic tabular data generation. In Will Synthetic Data Finally Solve the Data Access Problem?, 2025. URL https://openreview.net/forum?id=9FIOO09boS.

Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Alex Labach, Hamidreza Kamkari, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Anthony L. Caterini, and Maksims Volkovs. TabDPT: Scaling tabular foundation models on real data. In Advances in Neural Information Processing Systems (NeurIPS), 2025.

Markus Mueller, Kathrin Gruber, and Dennis Fok. Continuous diffusion for mixed-type tabular data. In International Conference on Learning Representations (ICLR), 2025.

Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 399–410, 2016.

Zhaozhi Qian, Bogdan-Constantin Cebere, and Mihaela van der Schaar. Synthcity: facilitating innovative use cases of synthetic data in different data modalities. arXiv preprint arXiv:2301.07573, 2023.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In Proceedings of the 42nd International Conference on Machine Learning (ICML), 2025.

Andrei Rimsa, Mark A. J. Song, and Luis E. Zárate. SCGaz — a synthetic formal context generator with density control for test and evaluation of FCA algorithms. In 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3464–3470. IEEE, 2013.

Taro Sakurai. On formal concepts of random formal contexts. Information Sciences, 578: 615–620, 2021.

Zineb Senane, Axel Karlsson, Lele Cao, Oleg Smirnov, Cheng Zhang, Sahar Asadi, Hedvig Kjellström, Gustav Eje Henter, and Ruibo Tu. Causality for tabular data synthesis: A high-order structure causal benchmark framework. In Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2026.

Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. TabDiff: a mixed-type diffusion model for tabular data generation. In International Conference on Learning Representations (ICLR), 2025.

Aivin V. Solatorio and Olivier Dupriez. REaLTabFormer: Generating realistic relational and tabular data using transformers. arXiv preprint arXiv:2302.02041, 2023.

Mihaela C. Stoian, Eleonora Giunchiglia, and Thomas Lukasiewicz. A survey on deep learning approaches for tabular data generation: Utility, alignment, fidelity, privacy, diversity, and beyond. Trans. Mach. Learn. Res., 2026. URL https://openreview.net/forum?id=RoShSRQQ67.

Paul Tiwald, Ivona Krchova, Andrey Sidorenko, Mariana Vargas Vieyra, Mario Scriminaci, and Michael Platzer. TabularARGN: A flexible and efficient auto-regressive framework for generating high-fidelity synthetic data. arXiv preprint arXiv:2501.12012, 2025.

Chaithra Umesh, Kristian Schultz, Manjunath Mahendra, Saptarshi Bej, and Olaf Wolkenhauer. Preserving logical and functional dependencies in synthetic tabular data. Pattern Recognit., 163:111459, 2025.

Chaithra Umesh, Kristian Seegel-Schultz, Manjunath Mahendra, Saptarshi Bej, and Olaf Wolkenhauer. Dependency-aware synthetic tabular data generation. Pattern Recognit., 179:113819, 2026.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS), volume 206 of PMLR, 2023.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In Advances in Neural Information Processing Systems (NeurIPS), 2019.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In International Conference on Learning Representations (ICLR), 2024.

Zilong Zhao, Aditya Kunar, Hiek Van der Scheer, Robert Birke, and Lydia Y. Chen. CTAB-GAN: Effective table data synthesizing. arXiv preprint arXiv:2102.08369, 2021.

A Corpus composition and excluded contexts

Of the 63 contexts, 15 come from the FCA contexts repository and 48 from KONECT. Within KONECT one family dominates: 21 contexts, 33% of the corpus, are bipartite wiki edit- networks, and they are very similar to each other. After clarification and reduction they range from 12×12 to 28×30, with densities of 0.04–0.10 and 15 to 38 intents, which is less than one order of magnitude of lattice size in a corpus that spans 5 to 49 152 intents. The family therefore adds contexts but little variety. The variety comes from the rest of KONECT (14 moreno_ animal and small-group social networks, 4 brunson_ affiliation networks and 9 contexts from 8 further collections) and from the curated contexts. Table 1 lists every context with its ground-truth measures. A median over contexts weights the edit- family by its share. We do not reweight, since every generator is scored on the same contexts, so that an over-represented family is over-represented for all of them and cannot favour one of them. However, it limits how far the effect sizes generalise (Section 6).

We excluded three kinds of prepared contexts. First, 3 KONECT season networks are degenerate. In a round-robin season every team plays every other team, so the adjacency matrix is the complete digraph without its diagonal. This is a contranominal scale, whose concept lattice is the Boolean lattice on the objects, with $2^{|G|}$ intents. These contexts have dimensions 18×18, 20×20 and 20×20 and 2^{18}, 2^{20} and 2^{20} intents, and a generator reproduces them by emitting a nearly complete matrix, so there is no structure to learn. Two of them are also byte-identical, the only such pair among the 74 prepared contexts. The criterion is a property of the lattice and not of the kind of network. A fourth season network, konect_league_be1_2016, is kept, because its schedule splits into playoff groups. At 16×16 its lattice has 49 152 intents instead of the 65 536 of a contranominal scale, which makes it the largest lattice in the corpus.

Second, 5 contexts duplicate others. A hash finds a repeated file but not a repeated context, since permuting the rows or columns of a context changes its file but not its lattice. We therefore also compared the prepared contexts up to a permutation of rows and columns, by canonical form, and found the following pairs of excluded and kept context: rwc_tealady $\cong$ konect_opsahl_southernwomen, rwc_driveconcepts_en $\cong$ rwc_driveconcepts_de, rwc_livingbeings_en $\cong$ rwc_livingbeings_de, rwc_missmarple_en $\cong$ rwc_missmarple_de and rwc_officesupplies_en $\cong$ rwc_officesupplies_de. 4 are one curated table prepared from its German and from its English edition, and 1 is a classic table that appears in both collections under different names. Of each pair we keep the lexicographically first name. As every quantity we report is invariant under isomorphism, a duplicate would add a second, identical vote to every median, sign test and bootstrap resample, and narrow the intervals without adding evidence.

Third, 3 KONECT contexts, prepared with size caps well above those of the corpus, could not be measured in full. Of the 8 430 measurement jobs implied by their 1 686 binarised matrices, 5 267 completed and 3 163 exceeded the budget, a coverage of 62.5%. Density, which needs no lattice, completed on 100% of the matrices, the pseudo-intent count on 25%. The size of the matrix does not predict this. konect_moreno_innovation is the largest of the three at 212×189 and completes best, at 84.8%, whereas konect_arenas_jazz, at 174×158, completes worst, at 50.6%, and has an order of magnitude more intents (8 074 against 828). The cost depends on the size of the lattice, which cannot be read off $|G|$ and $|M|$, so we validated the size caps by measurement. A larger budget would not recover these contexts either: the pseudo-intent count exceeds even the 1800 s budget for source contexts on all 3 source contexts, so there is no ground truth to score a generated table against.

Table 1: FCA properties of the source contexts, the ground truth that every generated table is compared with. A dash marks a measure that did not complete within the budget.

Context	Density	Intents	Pseudo-Int.	Meet-Rise	Join-Rise
konect_brunson_club_membership	0.267	62	73	0.338	0.515
konect_brunson_corporate_leadership	0.230	68	85	0.432	0.368
konect_brunson_south_africa	0.440	10	5	0.214	0.357
konect_brunson_southern_women	0.560	14	5	0.217	0.217
konect_contiguous_usa	0.078	70	329	0.441	0.455
konect_dimacs10_chesapeake	0.229	78	107	0.388	0.359
konect_dolphins	0.100	83	401	0.508	0.525
konect_edit_angwikisource	0.090	20	63	0.406	0.375
konect_edit_biwikibooks	0.056	30	173	0.472	0.434
konect_edit_bmwikibooks	0.054	31	184	0.426	0.407
konect_edit_bmwikiquote	0.043	31	245	0.473	0.473
konect_edit_chwikibooks	0.066	31	147	0.500	0.393
konect_edit_gnwikibooks	0.057	24	143	0.463	0.415
konect_edit_gotwikibooks	0.064	33	236	0.508	0.361
konect_edit_krwikiquote	0.104	15	47	0.417	0.417
konect_edit_kswikibooks	0.058	32	221	0.508	0.441
konect_edit_kswikiquote	0.052	30	182	0.431	0.373
konect_edit_mhwiktionary	0.049	34	239	0.452	0.468
konect_edit_miwikibooks	0.081	26	93	0.455	0.364
konect_edit_mywikibooks	0.057	30	173	0.481	0.463
konect_edit_nahwikibooks	0.052	30	202	0.472	0.453
konect_edit_quwikibooks	0.056	31	219	0.463	0.389
konect_edit_rmwikibooks	0.061	26	142	0.409	0.386
konect_edit_rnwiktionary	0.058	29	176	0.481	0.462
konect_edit_tkwikiquote	0.049	33	237	0.397	0.431
konect_edit_ttwikiquote	0.067	37	181	0.391	0.344
konect_edit_ugwikiquote	0.047	35	261	0.403	0.419
konect_edit_vowikiquote	0.055	38	283	0.441	0.368
konect_hiv	0.085	23	98	0.463	0.488
konect_iceland	0.120	39	96	0.472	0.333
konect_league_be1_2016	0.934	49152	1	–	–
konect_moreno_beach	0.244	395	260	0.394	0.364
konect_moreno_bison	0.499	1835	260	0.225	0.294
konect_moreno_cattle	0.433	417	183	0.404	0.342
konect_moreno_hens	0.484	205	93	0.299	0.364
konect_moreno_highschool	0.084	336	1460	0.371	0.458
konect_moreno_kangaroo	0.493	46	17	0.097	0.075
konect_moreno_lesmis	0.099	112	318	0.419	0.304
konect_moreno_rhesus	0.482	192	88	0.325	0.381
konect_moreno_sampson	0.583	498	129	0.283	0.330
konect_moreno_seventh	0.462	1719	384	0.288	0.277
konect_moreno_sheep	0.435	670	237	0.333	0.322
konect_moreno_taro	0.161	54	224	0.615	0.615
konect_moreno_train	0.132	89	164	0.436	0.366
konect_moreno_zebra	0.216	44	43	0.286	0.221
konect_opsahl_southernwomen	0.400	65	22	0.209	0.324
konect_ucidata_gama	0.371	45	35	0.388	0.408
konect_ucidata_zachary	0.210	34	54	0.328	0.328
rwc_animals_en	0.382	35	29	0.303	0.500
rwc_binary_relations_en	0.571	50	9	0.095	0.500
rwc_bodiesofwater_de	0.500	28	3	0.129	0.484
rwc_bodiesofwater_en	0.533	12	3	0.056	0.333
rwc_driveconcepts_de	0.700	24	5	0.240	0.040
rwc_famous_animals_en	0.467	13	7	0.238	0.286
rwc_forum_romanum_en	0.476	19	8	0.097	0.226
rwc_livingbeings_de	0.406	19	9	0.250	0.344
rwc_missmarple_de	0.381	13	5	0.238	0.571
rwc_music_en	0.635	163	18	0.110	0.613
rwc_newzealand_en	0.562	8	2	0.000	0.000
rwc_officesupplies_de	0.444	5	2	0.200	0.200
rwc_planets_en	0.429	12	10	0.278	0.278
rwc_seasoningplanner_de	0.207	532	551	0.353	0.492
rwc_triangles_en	0.469	18	7	0.129	0.355

B Generators and binarisation

Table 2 lists the 15 generators that produce samples, with the implementation and version that we ran. Each generator is fitted on one context at a time, with the last attribute as the target of the supervised interface (Section 3), and runs at the defaults of its implementation apart from the settings in the table. Where an implementation expects a declaration of discrete columns, we declare every attribute discrete. The only exception is the Gaussian variant of TabDDPM, which declares none, and this is the only difference between the two TabDDPM variants. The TabPFN models receive no declaration, TabICL decides for each column by its own rule, and our TabDPT sampler treats every attribute as discrete. As TabDPT has no generative interface, we sample from it autoregressively: in a random column order, each column is drawn from the distribution that TabDPT predicts given the columns generated so far.

Table 2: The generators that produce samples. Versions are those of the released lock file. Settings lists the values that our wrappers set explicitly, apart from seeds and the declaration of discrete columns (see text); everything else is the default of the implementation.

Generator	Implementation	Version	Settings
CTGAN (Xu et al., 2019)	SDV (Patki et al., 2016)	sdv 1.36.1, ctgan 0.12.1	—
TVAE (Xu et al., 2019)	SDV	sdv 1.36.1, ctgan 0.12.1	—
CTABGAN (Zhao et al., 2021)	reference code, modified	commit 73d4e31	100 epochs; no test split; mixture components scaled to the table
SynthyverseCTGAN	synthyverse	0.2.1	300 epochs
TabDDPM (Kotelnikov et al., 2023)	synthyverse	0.2.1	no discrete columns
TabDDPM_binary	synthyverse	0.2.1	—
TabARGN (Tiwald et al., 2025)	synthyverse, mostlyai-engine	0.2.1, 1.1.6	—
ARF (Watson et al., 2023)	arfpy	0.1.1	—
ForestDiffusion (Jolicoeur-Martineau et al., 2024)	reference code	1.0.6 (commit 818ac3b)	—
TabPFN (v2.5) (Grinsztajn et al., 2025)	tabpfn, tabpfn-extensions	8.0.8, 0.3.0	temperature 1
TabPFN3 (Grinsztajn et al., 2026)	tabpfn, tabpfn-extensions	8.0.8, 0.3.0	temperature 1
TabDPT (Ma et al., 2025)	tabdpt, our sampler	1.1.13	temperature 1; the whole context as the prompt
TabICL (Qu et al., 2025)	tabicl	2.1.1	temperature 1; at most 10 categories per column
Cointoss	conexp-clj (Hanika & Hirth, 2019)	2.7.0	—
Dirichlet (Felde & Hanika, 2019)	conexp-clj	2.7.0	—

The 3 generators with continuous output are binarised by density-original, a single global threshold chosen such that the density of the matrix is as close as possible to that of its source. The output of the other generators is already binary and is measured as it is (identity). Together, these two rules form the binarisation of the main text.

C Coverage and missing measurements

Table 3 shows what was measured at the binarisation of the main text: for each generator, the contexts it covers, the seeds and the number of binary matrices measured. The remainder of this appendix accounts for what is missing. To this end, the aggregation stage reconstructs every measurement that the design expects (for each context, registered generator, seed and measure), compares them with what is on disk, and assigns every missing measurement a reason by tracing it back through the pipeline, that is, whether a table was generated, whether it was binarised, and whether a measurement was written that can be parsed.

Table 3: Coverage of each generator at the binarisation of the main text: the number of source contexts and seeds, and the number of binary matrices that reached the measurement stage.

Generator	Category	Contexts	Seeds	Matrices
TABARGN	Autoregressive	61	10	610
CTABGAN	Deep (GAN/VAE)	50	10	500
CTGAN	Deep (GAN/VAE)	63	10	630
SYNTHYVERSECTGAN	Deep (GAN/VAE)	50	10	500
TVAE	Deep (GAN/VAE)	63	10	630
TABDDPM	Diffusion	63	10	630
TABDDPM_BINARY	Diffusion	63	10	630
TABDPT	Foundation (ICL)	63	10	630
TABICL	Foundation (ICL)	63	10	630
TABPFN	Foundation (ICL)	63	10	630
TABPFN3	Foundation (ICL)	63	10	630
COINTOSS	Random Baseline	63	10	630
DIRICHLET	Random Baseline	63	10	630
ARF	Tree Ensemble	63	10	630
FORESTDIFFUSION	Tree Ensemble	63	10	630

Table 4 reports the result in two units, which are not summed. A cell-level absence is a (context, generator, seed) combination without a table, for which no measure can be named. An artefact-level absence names the missing measurement. The absences by design say nothing about a generator: stub-sampler covers the 3 generators that we did not train, and sampler-degenerate covers inputs that a method declines by design, such as SMOTE's empty balancing set.

If the failures were concentrated in some generators, these would be scored on a different part of the corpus. Table 5 breaks the failures down by generator. 461 measurements are missing, 176 of them on one of the two counts, which is 1.0% of the count measurements expected. These are spread over 13 of the 15 generators but concentrated on 3 contexts, with 139 on konect_moreno_highschool alone, where the enumeration of pseudo-intents (Distel & Sertkaya, 2011) exceeds the budget for 13 of the 15 generators. Therefore, a measure goes missing on a few expensive contexts for many generators at once, and no generator is scored on an easier subset than the others.

However, one asymmetry remains. 49 of the 176 gaps fall on the two baselines, 29 of them outside konect_moreno_highschool. A table filled at random at the source density has a much larger canonical base than the context it imitates, so the baselines exceed the budget first. The pseudo-intent count is also the count on which the baselines are further from the source, so where it is missing they are scored on the intent count alone, which favours them. The asymmetry therefore favours C3. Restricting the comparison to the contexts that every generator covers limits this effect but does not remove it, since the largest lattices still cost the baselines a measure.

The remaining 285 gaps are rise-rate measurements. They do not enter the count score, but they thin out the check of Section 4.3, most of all for TABARGN, with 76 gaps, which is also the outlier of that check. Its rise-rate score therefore rests on fewer contexts than those of the other generators.

Table 4: Skip accounting at the binarisation of the main text: every measurement the design expects but the run does not contain, with the reason it is absent. By-design absences follow from the experimental design; gaps are failures and are broken down per generator in Table 5.

Reason	Class	Cells	Artefacts	Gen.	Meaning
stub-sampler	By design	1,890	0	3	Generator needs a pre-training stage we could not reproduce; not run.
sampler-degenerate	By design	630	0	1	Method undefined on the input (single class, or fewer than two rows per class).
sample-missing	Gap	910	0	4	Generator produced no table, with no upstream reason recorded.
scale-missing	Gap	0	0	0	Table produced, binarised artefact absent.
measure-failed	Gap	0	461	14	Binarised artefact present, measurement did not finish within its budget.
measure-empty	Gap	0	0	0	Measurement file written but held no usable value.

Cells count whole (context, generator, seed) combinations whose absence cannot be attributed to one measure; artefacts count individual missing measurements. The two are different units and are not summed. By-design absences total 2,520, actionable gaps 1,371.

Table 5: Actionable gaps per generator at the binarisation of the main text. By-design absences (Table 4) are excluded: they follow from the design rather than from a generator failing.

Generator	Category	sample-missing	measure-failed	measure-empty	Total
CDTD	Diffusion	630	0	0	630
SYNTHYVERSECTGAN	Deep (GAN/VAE)	130	44	0	174
CTABGAN	Deep (GAN/VAE)	130	28	0	158
TABARGN	Autoregressive	20	76	0	96
TABDDPM	Diffusion	0	59	0	59
COINTOSS	Random Baseline	0	57	0	57
CTGAN	Deep (GAN/VAE)	0	50	0	50
DIRICHLET	Random Baseline	0	49	0	49
TABPFN	Foundation (ICL)	0	22	0	22
ARF	Tree Ensemble	0	20	0	20
TABPFN3	Foundation (ICL)	0	18	0	18
TABDPT	Foundation (ICL)	0	12	0	12
TABDDPM_BINARY	Diffusion	0	10	0	10
TABICL	Foundation (ICL)	0	10	0	10
FORESTDIFFUSION	Tree Ensemble	0	6	0	6

The sample-missing gaps have two causes on this corpus. Some generators decline contexts with fewer rows than they support (CTAB-GAN and synthyverse's CTGAN below ten rows, TabARGN on the smallest contexts), and CDTD requires a continuous column and therefore fails on every context. The pipeline cannot distinguish either case from a crash without encoding the preconditions of each library, so the reason code remains conservative. The ranking is not affected, since it is computed on the contexts that every generator covers.

D The count comparison in full

This appendix contains the tables behind Sections 4.1 and 4.2. Table 6 gives the median signed deviation of each generator on each of the five measures at the source density, including the three measures that Figure 2 does not show. Table 8 gives the structural-fidelity score with its cluster-bootstrap interval over contexts, the interval of the rank of each generator, recomputed within each of the 10000 replicates, and the two components of the score. Table 9 gives all 26 comparisons of a baseline with a learned generator, each on the contexts both cover, with a paired bootstrap interval for the difference and an exact sign test for its direction, Holm-corrected over the whole family. In addition, each row reports the smallest p that its number of shared contexts allows, so that an inconclusive verdict can be checked against what was attainable. Finally, Table 7 gives the mean coefficient of variation of the measurements of each generator across the 10 seeds. The least variable generator on both counts is TABARGN (0.108 on closed sets, 0.078 on pseudo-intents), which is also the generator with the largest loss of closed sets (Section 4.1), so its tables differ less from each other as well as from the source. No generator is near-constant, since apart from density, which the binarisation fixes for a generator with non-binary output, the smallest mean coefficient of variation is 0.078, so every cell averages over 10 different tables.

Table 6: Directional bias of each sampler versus the original contexts, on the contexts covered by every sampler and under the same scaler rule as Table 8 (density-original, or identity where the generator emits 0/1); the density column is therefore the residual of that thresholding, not an unconstrained bias. Units differ by measurement scale: the count measures (intents, pseudo-intents) report the median $\log_2((\text{generated}+1)/(\text{original}+1))$, so -1 means half as many; the bounded proportions (density, meet- and join-rise rate) report the median additive difference generated − original. Negative = undershoots the source (simpler structure), positive = overshoots; 0 = faithful.

Sampler	Category	Density	Intents	Pseudo-Int.	Meet-Rise	Join-Rise
TABARGN	Autoregressive	-0.05	-3.00	-2.10	-0.41	-0.37
CTABGAN	Deep (GAN/VAE)	+0.00	-0.30	-0.75	-0.03	+0.03
CTGAN	Deep (GAN/VAE)	-0.01	-0.31	-0.58	-0.01	+0.05
SYNTHYVERSECTGAN	Deep (GAN/VAE)	-0.00	-0.25	-0.59	-0.02	+0.03
TVAE	Deep (GAN/VAE)	-0.06	-2.91	-2.41	-0.29	-0.29
TABDDPM	Diffusion	+0.09	-0.49	-0.69	+0.01	-0.02
TABDDPM_BINARY	Diffusion	-0.00	-0.39	-0.71	-0.05	+0.01
TABDPT	Foundation (ICL)	+0.01	-0.30	-0.67	-0.02	+0.00
TABICL	Foundation (ICL)	+0.01	-0.30	-0.82	-0.01	-0.01
TABPFN	Foundation (ICL)	+0.00	-0.36	-0.92	-0.04	+0.04
TABPFN3	Foundation (ICL)	+0.00	-0.30	-0.80	-0.03	+0.04
COINTOSS	Random Baseline	+0.00	-0.09	-0.25	+0.03	+0.03
DIRICHLET	Random Baseline	+0.00	-0.14	-0.14	+0.02	+0.02
ARF	Tree Ensemble	+0.00	-0.19	-0.55	-0.01	+0.03
FORESTDIFFUSION	Tree Ensemble	+0.04	-0.18	-1.41	-0.03	-0.00

Table 7: Seed stability: mean coefficient of variation (std/mean) of each measure across random seeds at the binarisation of the main text, averaged over contexts. Lower = more reproducible output across seeds. For a generator with non-binary output, this binarisation fixes the density to that of the source, so its density varies little by construction.

Sampler	Category	Density	Intents	Pseudo-Int.	Meet-Rise	Join-Rise
TABARGN	Autoregressive	0.115	0.108	0.078	0.617	0.642
CTABGAN	Deep (GAN/VAE)	0.064	0.214	0.214	0.178	0.147
CTGAN	Deep (GAN/VAE)	0.108	0.213	0.188	0.382	0.377
SYNTHYVERSECTGAN	Deep (GAN/VAE)	0.136	0.256	0.205	0.167	0.135
TVAE	Deep (GAN/VAE)	0.584	0.258	0.102	0.889	0.806
TABDDPM	Diffusion	0.087	0.164	0.140	0.151	0.188
TABDDPM_BINARY	Diffusion	0.059	0.143	0.149	0.303	0.291
TABDPT	Foundation (ICL)	0.102	0.214	0.187	0.253	0.321
TABICL	Foundation (ICL)	0.125	0.241	0.206	0.360	0.434
TABPFN	Foundation (ICL)	0.000	0.166	0.178	0.311	0.281
TABPFN3	Foundation (ICL)	0.000	0.166	0.203	0.331	0.308
COINTOSS	Random Baseline	0.124	0.195	0.207	0.227	0.204
DIRICHLET	Random Baseline	0.086	0.195	0.190	0.214	0.230
ARF	Tree Ensemble	0.134	0.227	0.204	0.340	0.319
FORESTDIFFUSION	Tree Ensemble	0.131	0.248	0.168	0.309	0.382

Table 8: Structural fidelity with uncertainty. Score is the median $|\log_2((v+1)/(v_0+1))|$ over the two counts at the source density, on the 50 contexts every generator covers (lower is closer; 1 is off by a factor of two); equal scores share a rank. The intervals are 95% percentile intervals from a cluster bootstrap that resamples the contexts (with all their seeds and measures) 10000 times; Rank CI is the interval of each generator's rank across replicates, computed within each replicate. Intents and Pseudo-Int. are the score's two components, All ctx. the score over each generator's full coverage (up to 63 contexts), and Dens. gap the median absolute density difference that remains after binarisation (0.000 for the generators with a threshold to choose).

Rank	Sampler	Category	Score	95% CI	Rank CI	Intents	Pseudo-Int.	All ctx.	Dens. gap
1	DIRICHLET	Random Baseline	0.419	[0.354, 0.510]	[1, 6]	0.30	0.70	0.38	0.007
2	ARF	Tree Ensemble	0.465	[0.384, 0.567]	[1, 8]	0.32	0.80	0.44	0.010
3	TABPFN3	Foundation (ICL)	0.465	[0.401, 0.541]	[1, 6]	0.36	0.87	0.44	0.000
4	TABDPT	Foundation (ICL)	0.485	[0.418, 0.555]	[1, 8]	0.34	0.73	0.45	0.012
5	TABDDPM_BINARY	Diffusion	0.512	[0.445, 0.585]	[2, 11]	0.39	0.71	0.49	0.008
6	TABICL	Foundation (ICL)	0.516	[0.466, 0.585]	[3, 11]	0.35	0.85	0.49	0.017
7	CTABGAN	Deep (GAN/VAE)	0.518	[0.415, 0.608]	[3, 10]	0.40	0.91	0.52	0.000
8	COINTOSS	Random Baseline	0.534	[0.445, 0.636]	[2, 12]	0.32	0.99	0.49	0.009
9	TABPFN	Foundation (ICL)	0.542	[0.460, 0.628]	[4, 12]	0.39	0.95	0.49	0.000
10	SYNTHYVERSECTGAN	Deep (GAN/VAE)	0.564	[0.475, 0.695]	[5, 12]	0.39	0.96	0.56	0.011
11	CTGAN	Deep (GAN/VAE)	0.585	[0.484, 0.705]	[6, 13]	0.45	0.94	0.54	0.012
12	TABDDPM	Diffusion	0.606	[0.545, 0.686]	[8, 12]	0.52	0.73	0.54	0.096
13	FORESTDIFFUSION	Tree Ensemble	0.695	[0.600, 0.823]	[11, 13]	0.37	1.41	0.58	0.044
14	TVAE	Deep (GAN/VAE)	2.636	[2.303, 2.824]	[14, 15]	2.91	2.41	2.23	0.061
15	TABARGN	Autoregressive	2.787	[2.463, 3.000]	[14, 15]	3.13	2.51	2.51	0.056

Table 9: Each trivial baseline against each learned generator, on the contexts both cover. Δ is the baseline's structural fidelity minus the generator's, so negative favours the baseline; the interval is a paired cluster bootstrap over those contexts. Wins counts contexts where the baseline is closer to the source, and p_{sign} is the exact two-sided sign test on that count – distribution-free, and answering whether where the interval answers by how much. p_{Holm} corrects over all 26 comparisons in this pre-specified family. The family was fixed before any score was inspected; no comparison here was selected for its outcome.

Baseline	Generator	N	Δ	95% CI	Wins	p_{sign}	p_{Holm}	Verdict
COINTOSS	TABARGN	61	−2.024	[−2.323, −1.165]	46/61	$< 10^{-4}$	0.002	baseline better
COINTOSS	TVAE	61	−1.743	[−2.154, −1.394]	54/61	$< 10^{-4}$	$< 10^{-4}$	baseline better
COINTOSS	FORESTDIFFUSION	62	−0.100	[−0.201, 0.022]	45/62	0.000	0.010	baseline better
COINTOSS	CTGAN	62	−0.055	[−0.132, 0.028]	34/62	0.526	1.000	equivalent
COINTOSS	TABDDPM	63	−0.055	[−0.137, 0.030]	34/63	0.615	1.000	equivalent
COINTOSS	SYNTHYVERSECTGAN	50	−0.030	[−0.136, 0.076]	30/50	0.203	1.000	equivalent
COINTOSS	TABICL	63	−0.009	[−0.100, 0.090]	29/63	0.615	1.000	equivalent
COINTOSS	TABPFN	62	−0.009	[−0.103, 0.084]	37/62	0.162	1.000	equivalent
COINTOSS	TABDDPM_BINARY	63	−0.003	[−0.114, 0.092]	27/63	0.314	1.000	equivalent
COINTOSS	CTABGAN	49	+0.016	[−0.085, 0.140]	30/49	0.152	1.000	equivalent
COINTOSS	TABDPT	62	+0.039	[−0.059, 0.138]	27/62	0.374	1.000	equivalent
COINTOSS	ARF	59	+0.042	[−0.041, 0.121]	29/59	1.000	1.000	equivalent
COINTOSS	TABPFN3	63	+0.046	[−0.043, 0.137]	33/63	0.801	1.000	equivalent
DIRICHLET	TABARGN	60	−2.126	[−2.431, −1.263]	53/60	$< 10^{-4}$	$< 10^{-4}$	baseline better
DIRICHLET	TVAE	60	−1.850	[−2.248, −1.486]	57/60	$< 10^{-4}$	$< 10^{-4}$	baseline better
DIRICHLET	FORESTDIFFUSION	62	−0.206	[−0.295, −0.081]	48/62	$< 10^{-4}$	0.000	baseline better
DIRICHLET	CTGAN	60	−0.162	[−0.229, −0.074]	42/60	0.003	0.048	baseline better
DIRICHLET	TABDDPM	63	−0.162	[−0.231, −0.084]	44/63	0.002	0.042	baseline better
DIRICHLET	SYNTHYVERSECTGAN	50	−0.145	[−0.246, −0.044]	35/50	0.007	0.112	equivalent
DIRICHLET	TABICL	61	−0.116	[−0.200, −0.021]	44/61	0.001	0.015	baseline better
DIRICHLET	TABPFN	63	−0.116	[−0.200, −0.024]	42/63	0.011	0.167	equivalent
DIRICHLET	TABDDPM_BINARY	61	−0.110	[−0.202, −0.028]	38/61	0.072	0.938	equivalent
DIRICHLET	CTABGAN	50	−0.099	[−0.196, 0.019]	31/50	0.119	1.000	equivalent
DIRICHLET	TABDPT	63	−0.068	[−0.161, 0.027]	41/63	0.023	0.316	equivalent
DIRICHLET	ARF	60	−0.065	[−0.138, 0.010]	36/60	0.155	1.000	equivalent
DIRICHLET	TABPFN3	61	−0.061	[−0.142, 0.026]	41/61	0.010	0.158	equivalent

Equivalent means the whole interval falls inside ± 0.5 – the gap is bounded below what we would call a substantive difference – not that it is zero. Inconclusive means neither: the data neither separates the two nor bounds their difference.

E The rise-rate check

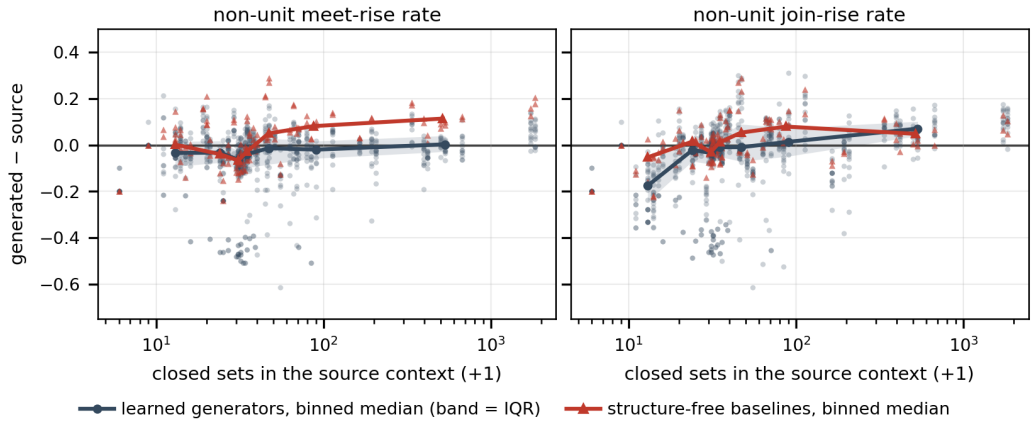

Figure 5: Figure 3 for the two rise rates: the difference of each (context, generator) cell from the source rate against the number of closed sets in the source, with binned medians for the learned generators (band: interquartile range) and the baselines. Contexts whose source rise rates exceed the measurement budget have no points.

The rise rates address the objection that the count score is too coarse to distinguish a table filled at random from the output of a learned generator. As bounded proportions they cannot be pooled with the counts (Section 3), but they are pooled with each other and put through the same procedure, with the same 26 pairs fixed in advance, the same Holm correction and the same bootstrap seed. Section 4.3 reports the outcome: 8 comparisons survive the correction, 4 in favour of a baseline and 4 in favour of a learned generator, whereas all 9 comparisons that the counts decide favour the baseline.

We use the rise rates as a check and not as a second ranking, for two reasons. First, they are undefined on konect_league_be1_2016, the largest lattice in the corpus with 49 152 intents, whose rise rates exceed the measurement budget, and a measure that fails where the structure is richest cannot rank the generators on its own. Secondly, they measure something different, as the rank correlation of the two orderings is only 0.24. For the same reason they are a useful check, since a measure this different could have shown the count result to be an artefact of the count score. This is not the case. Both scores find the loss, with TABARGN as the outlier on both, and they disagree only on comparisons with Cointoss that the counts find equivalent, which is why C3 holds against Cointoss on the counts only.

F Worked example

The measures of Section 3 are counts and proportions over a concept lattice, and every result in Section 4 is a summary over contexts. This appendix shows a single context that is small enough for its lattice to be drawn. For each generator we show one seed, chosen by a fixed rule: the seed whose number of closed sets is closest to the median of the generator over its seeds, and the lowest such seed on a tie. Thus, each panel shows a typical output of the generator rather than its best or its worst, and every generator is shown.

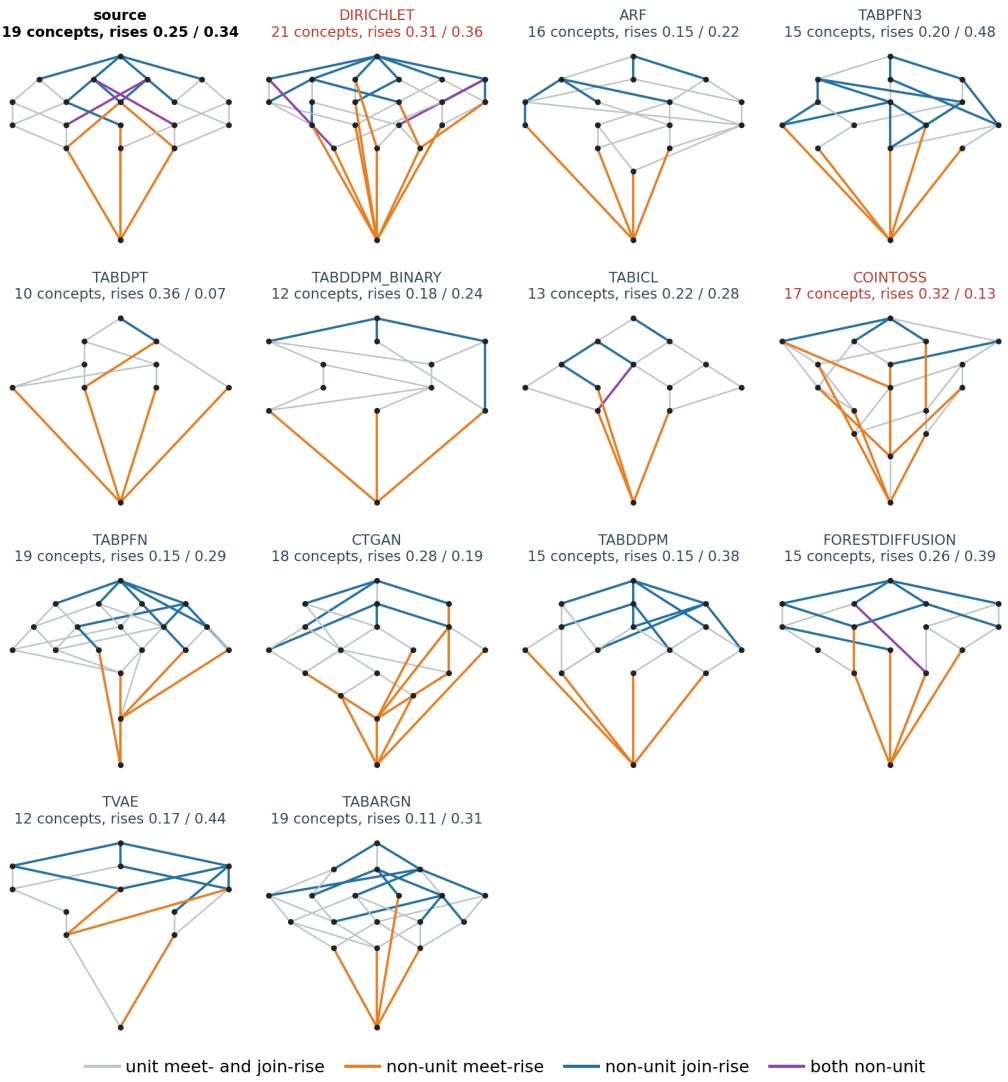

Figure 6: Concept lattice of the "living beings and water" context (Ganter & Wille, 1999), the textbook example of FCA, and of one output of each generator at the source density, in the order of the structural-fidelity score. Edges are the covering pairs, coloured by whether the pair has a non-unit meet-rise, a non-unit join-rise, both or neither; the two numbers in each title are the non-unit meet- and join-rise rates, that is, the share of edges of each kind.

G Reproducibility details

This appendix gives the mechanism behind the claims made in the reproducibility statement.

Sampling is controlled by a single --seed argument. The seed of an individual sample is derived deterministically from the triple (global seed, context name, sample index), so it does not depend on the order in which contexts or samples are processed, and the derivation is the same in the Python and the Clojure samplers. Before each sample, the harness seeds random, NumPy and PyTorch. When a seed is given, it additionally enforces deterministic algorithms, cuDNN behaviour and the cuBLAS workspace configuration, so that the generators that run on a GPU are reproducible as well. Generators with an explicit seed parameter receive a seed drawn from a separate random stream. This stream is isolated from the global one, because several dependencies draw from the global NumPy generator when they are first imported in a process, which would otherwise make the first sample of a run differ from later ones.

To verify this, a dedicated harness samples a fixed context twice with the same seed and compares the outputs bit by bit, and all generators evaluated here pass. The first generator run in a process pays a one-time CUDA initialisation, whose transient state can make a cold run differ from a later warm run in the same process. This only affects testing within one process, since a re-run of the pipeline always starts cold, which we confirmed by comparing two runs in separate processes. The harness therefore discards one warm-up run before the two runs it compares.

All dependencies are pinned by a lock file across isolated environments, one per generator family and one each for the analysis and for FCA, and the few patches that upstream packages require are applied by an idempotent install script. The two TabPFN generators (Grinsztajn et al., 2025; 2026) load pretrained checkpoints that their authors distribute under non-commercial licences, and TabICL loads a pretrained checkpoint as well. None of these checkpoints is part of the released artefacts. Instead, the pipeline downloads them from the public model repositories of their authors at a pinned revision and accepts a file only if its SHA-256 digest equals that of the copy used for the reported runs. TabDPT downloads its weights at first use, and the revision served for the reported runs was not recorded.

The source contexts come from the FCA contexts repository (FCA Tools Working Group, 2024) and from KONECT (Kunegis, 2013), whose terms permit redistribution for research purposes, and we redistribute the prepared contexts with attribution to both. The generators are used under their own licences, at the versions listed in Appendix B.

The two rise rates require the concept lattice, and enumerating pseudo-intents is hard, so a small number of matrices cannot be measured within a fixed wall-clock budget. We use two budgets on purpose. A generated matrix is measured with a limit of 60 s per measure, and a source context with a limit of 1800 s. Generated matrices outnumber source contexts by orders of magnitude, so the limit on generated matrices is the one that bounds the total compute, and it can be set where a runaway measurement is cut off. A source context, in contrast, is the ground truth that its generated matrices are scored against, and a context without ground truth leaves the corpus (Appendix A). Since a guard against runaway compute should not decide which contexts are in the corpus, the limit for source contexts is set such that the measurement, not the schedule, is the constraint. Cells that exceed either budget are not dropped silently. Instead, the aggregation stage reconstructs all expected (context, generator, seed, measure) cells and records every missing one with a reason code. The codes distinguish absences by design (a stub generator was not run, an oversampling method had no rows to add) from actual failures (a sample, a binarisation or a measurement that failed). Appendix C gives the resulting accounting.