MEDM-GEN: A Pattern-Guided, Constraint-Aware Dataset Generator Framework for Reproducible Data Mining Research

Văn Hà Minh Quân
Independent Researcher
fhhdhysd@gmail.com

Abstract

Synthetic dataset generation has become essential for data mining research because real-world datasets are often scarce, privacy-sensitive, weakly annotated, biased, or unsuitable for controlled algorithmic benchmarking. Existing generators can synthesize tabular records, images, sequences, transactions, graphs, and time series using statistical models, oversampling, generative adversarial networks, variational autoencoders, normalizing flows, diffusion models, and domain-specific simulators. However, most generators are optimized either for visual/statistical realism, data augmentation, or privacy-preserving release; they are rarely designed as *research instruments* for data mining, where researchers need controllable ground-truth patterns, known difficulty factors, reproducible benchmarks, and systematic stress tests for algorithms. This paper consolidates two complementary survey streams: general synthetic data generation and data-mining-oriented dataset generation frameworks. From this synthesis, we identify a central gap: the lack of a unified, extensible, and pattern-aware generator that can produce benchmark datasets across data mining tasks while explicitly controlling pattern structure, noise, sparsity, imbalance, drift, constraints, privacy, and evaluation difficulty. To address this gap, we propose MEDM-GEN, a Q1-style framework for data mining dataset generation based on four principles: (i) a declarative dataset specification language, (ii) a ground-truth pattern ledger, (iii) modular task-specific generators, and (iv) closed-loop evaluation and repair. The proposed framework supports transactional, sequential, tabular, temporal, graph, and hybrid data mining settings. We formalize the problem, survey representative algorithms and tools, define the research gap, present the system architecture, provide generation and feedback algorithms, and design an experimental protocol for evaluating fidelity, utility, controllability, privacy, and reproducibility.

Keywords: Synthetic data generation; data mining benchmark; pattern mining; SPMF; dataset generator; reproducibility; controllable generation; privacy-preserving data mining.

1 Introduction

Data mining research depends heavily on datasets. A new frequent itemset mining algorithm, sequential pattern mining method, graph mining technique, clustering algorithm, or classification approach is usually evaluated by measuring runtime, memory usage, pattern recovery, prediction performance, robustness to noise, and scalability on benchmark datasets. However, real-world datasets have three structural limitations. First, they do not provide full control over the hidden generative process. Second, they rarely include complete ground-truth patterns. Third, they cannot easily be modified to isolate the impact of variables such as sparsity, density, long-tail frequency, drift, class imbalance, pattern overlap, and noise.

Synthetic data generation offers a partial solution. Traditional approaches such as bootstrapping, rule-based simulation, probabilistic models, copulas, Bayesian networks, and oversampling methods such as SMOTE create additional data under explicit statistical assumptions [1, 2, 8]. Deep generative models such as GANs, VAEs, normalizing flows, diffusion models, and autoregressive models learn complex distributions and can generate high-fidelity samples [3, 4, 5, 6]. Toolkits such as DataSynthesizer, SDV, SynthCity, synthpop, CTGAN, TVAE, TimeGAN, and SPMF provide usable implementations for specific data types or tasks [8, 10, 9, 11, 12, 13].

Nevertheless, the available ecosystem is not yet sufficient for a data mining framework that needs benchmark-grade control. General-purpose synthetic data systems aim to mimic real data distributions. Image and tabular generative models aim to improve sample realism. Privacy-preserving generators aim to publish useful approximations of sensitive data. Pattern mining generators, including generators distributed with SPMF, are useful for creating transaction and sequence databases but typically offer limited control over high-order dependencies, rare patterns, cross-task datasets, reproducibility metadata, and systematic difficulty calibration.

This paper proposes MEDM-GEN, a data generator framework for data mining research. The key idea is to shift the objective from *realism-only generation* to *benchmark-aware generation*. Instead of asking only whether a synthetic dataset looks similar to a real dataset, MEDM-GENasks whether a dataset can answer controlled research questions:

- Can the target algorithm recover known embedded patterns?
- How does performance change as sparsity, noise, imbalance, or pattern overlap increases?
- Can different algorithms be compared under identical generative conditions?
- Can a dataset be regenerated exactly from a specification file?
- Can synthetic data preserve utility while controlling privacy leakage?

1.1 Contributions

This paper makes five contributions.

1. It consolidates a broad survey of synthetic data generation methods with a data-mining-specific survey of dataset generators and benchmarking tools.
2. It defines the research gap separating general synthetic data generation from controllable data mining benchmark

generation.

3. It proposes MEDM-GEN, a modular framework combining a declarative specification layer, pattern ledger, task-specific generation kernels, constraint repair, privacy control, and closed-loop evaluation.
4. It formalizes the generation problem as a constrained multi-objective optimization problem balancing fidelity, utility, controllability, diversity, privacy, and reproducibility.
5. It provides algorithms and an evaluation protocol suitable for a Q1-level research paper.

2 Background and Survey

Synthetic data generation can be categorized by modeling assumption, target data type, controllability, and intended use. Table 1 summarizes the main families relevant to data mining.

2.1 Classical Statistical and Rule-Based Methods

Classical approaches include sampling from known distributions, bootstrapping, probabilistic graphical models, copulas, and rule-based engines. They are computationally efficient and interpretable, but their realism is limited by modeling assumptions. DataSynthesizer learns Bayesian network approximations and can add differential privacy noise [8]. synthpop generates synthetic microdata using sequential regression and tree-based synthesis [9]. These methods are useful for tabular data but not sufficient for general data mining benchmarks involving frequent patterns, temporal dependencies, graph structures, and controlled algorithmic difficulty.

2.2 Oversampling and Data Augmentation

SMOTE is a classic oversampling method for imbalanced classification. It interpolates minority-class instances with nearest neighbors to create synthetic samples [2]. Although effective for class imbalance, SMOTE is not a general-purpose dataset generator. It does not define global ground-truth patterns, cannot directly generate transaction/sequence/graph structures, and cannot control benchmark difficulty beyond local class density.

2.3 Deep Generative Models

GANs introduced adversarial training between a generator and a discriminator [3]. CTGAN adapts GANs to tabular data through conditional generation and mode-specific normalization [11]. VAEs learn probabilistic latent representations [4]. Diffusion models generate samples by learning a reverse denoising process [6]. These models are powerful, especially in images and high-dimensional domains, but for data mining benchmarks they have three weaknesses: weak interpretability of embedded structures, limited exact constraint satisfaction, and limited reproducibility of known ground-truth patterns.

2.4 Sequential, Temporal, and Graph Generators

TimeGAN combines adversarial and supervised objectives to preserve temporal dynamics in time-series generation [12]. Graph generators such as GraphRNN and NetGAN generate graph structures from learned graph distributions [17, 18]. These methods address important data mining modalities but are usually optimized independently rather than integrated into a unified benchmark generator.

2.5 Frameworks and Toolkits

SDV provides a practical ecosystem for tabular, relational, and sequential synthetic data generation [10]. SynthCity provides a plugin-based framework emphasizing tabular synthesis, privacy, and evaluation [19]. SPMF is an open-source data mining library containing numerous pattern mining algorithms and some synthetic data generators for transactions and sequences [13]. These tools are important, but none fully provides a universal data-mining-oriented generator with a declarative specification, ground-truth pattern ledger, difficulty calibration, privacy control, and closed-loop benchmark evaluation.

3 Research Gap

The survey reveals a gap between *synthetic data realism* and *data mining benchmark usefulness*. We define five missing capabilities.

3.1 Gap 1: Lack of Ground-Truth Pattern Ledgers

Many synthetic datasets are evaluated only by distributional similarity or downstream ML performance. For data mining, however, researchers need to know which patterns were embedded. For example, a frequent itemset mining benchmark should record the true itemsets, support values, correlations, noise injection, and overlap among patterns. A sequence mining benchmark should record temporal motifs, gap constraints, sequence length distributions, and drift events.

3.2 Gap 2: Weak Difficulty Control

Existing generators usually expose size and basic distribution parameters, but data mining difficulty depends on deeper properties: pattern overlap, density, support threshold sensitivity, class separability, noise ratio, long-tail skewness, concept drift, graph modularity, temporal irregularity, and rare-event frequency.

3.3 Gap 3: Single-Modality Fragmentation

Tabular, sequence, transaction, graph, and time-series generators are often separate. Modern data mining workflows increasingly combine modalities. A framework should allow hybrid specifications such as transactional records with temporal timestamps, graph nodes with tabular attributes, or event sequences with class labels.

3.4 Gap 4: Limited Reproducibility Metadata

Many synthetic generation pipelines do not produce a complete machine-readable record of generation assumptions, random seeds, embedded structures, evaluation metrics, and

Table 1: Taxonomy of synthetic data generation methods relevant to data mining research.

Family	Typical data	Strengths	Limitations for data mining benchmarks	Examples
Random sampling and bootstrapping	Numeric, tabular	Simple, scalable, preserves marginal statistics	Weak novelty, weak control over hidden structure, possible record reuse	Bootstrap [1]
Rule-based generation	Structured, transactional, simulation	Strong validity, explicit constraints, interpretable	Requires hand-crafted rules; may miss complex correlations	Faker, simulators
Probabilistic models	Tabular, relational	Captures dependency assumptions; interpretable; efficient	Model misspecification; limited high-order structure	Bayesian networks, copulas [8]
Oversampling	Imbalanced tabular classification	Improves minority-class learning; easy to use	Not a full data generator; local interpolation can create unrealistic points	SMOTE [2]
GAN-based models	Image, tabular, sequence	High-fidelity synthesis; flexible	Instability, mode collapse, privacy leakage, weak explicit pattern control	GAN, CTGAN [3, 11]
VAE-based models	Image, tabular, sequence	Stable training, latent structure	Lower sharpness/fidelity; weak exact constraint control	VAE, TVAE [4, 11]
Flow-based models	Continuous tabular, image	Exact likelihood, invertible sampling	Architectural constraints; difficult mixed data handling	RealNVP, Glow [5]
Diffusion models	Image, audio, emerging tabular	High sample quality, stable training	Slow sampling; high compute; not benchmark-structure aware	DDPM [6]
Autoregressive models	Text, sequence, time series	Strong sequential dependency modeling	Slow generation; exposure bias; privacy risk	Transformers, TimeGAN [7, 12]
Pattern-mining generators	Transaction, sequence	Directly useful for itemset/sequence mining; scalable	Usually narrow task scope; limited cross-modal and difficulty control	SPMF generators [13]

transformation history. This makes exact benchmark reproduction difficult.

3.5 Gap 5: Privacy and Utility Are Not Integrated with Benchmark Control

Privacy-preserving methods such as PrivBayes, DP-GAN, and PATE-GAN address leakage but often reduce utility [14, 15, 16]. They usually do not integrate privacy budgets with pattern recovery difficulty or benchmark calibration.

4 Problem Definition

Let $\mathcal{S}$ be a dataset specification containing schema, data type, target size N, constraints, task type, random seed, desired ground-truth patterns, privacy requirements, and difficulty parameters. Let $\mathcal{D}_r$ be an optional real seed dataset and $\mathcal{X}$ be the generated dataset. Let $\mathcal{P}^*$ denote the ground-truth pattern ledger to be embedded in the generated data.

The goal is to generate:

$$(\mathcal{X}, \mathcal{P}^*, \mathcal{M}) = \mathcal{G}(\mathcal{S}, \mathcal{D}_r; \theta),$$

where $\mathcal{M}$ is metadata sufficient for reproducibility.

We define the objective as:

$$\max_{\theta} \quad \alpha F(\mathcal{X}, \mathcal{D}_r) + \beta U(\mathcal{X}) + \gamma C(\mathcal{X}, \mathcal{P}^*) + \eta R(\mathcal{M}) - \lambda V(\mathcal{X}) - \rho P(\mathcal{X}, \mathcal{D}_r),$$

subject to:

$$\mathcal{X} \models \mathcal{C}, \quad |\mathcal{X}| = N, \quad \epsilon(\mathcal{X}, \mathcal{D}_r) \leq \epsilon_{\max}.$$

Here, F measures statistical fidelity to a real or specified distribution, U measures downstream utility, C measures controllability and pattern recoverability, R measures reproducibility completeness, V measures constraint violations, and P measures privacy risk. If no real dataset exists, F is computed against the declared distributional specification rather than $\mathcal{D}_r$.

4.1 Task-Specific Control Variables

For frequent itemset mining, control variables include transaction count, item universe size, average transaction length, support distribution, item correlation, planted itemsets, and noise. For sequential pattern mining, they include sequence length, event alphabet size, gap distribution, motif recurrence, and drift. For classification, they include class overlap, imbalance ratio, label noise, and feature relevance. For clustering, they include cluster shape, separation, density, and outlier rate. For graph mining, they include node count, edge density, community structure, motifs, and attribute correlation.

5 Proposed Framework

MEDM-GENis a modular dataset generation framework designed around six layers: specification, pattern ledger, generator kernel, constraint engine, evaluation engine, and feedback controller.

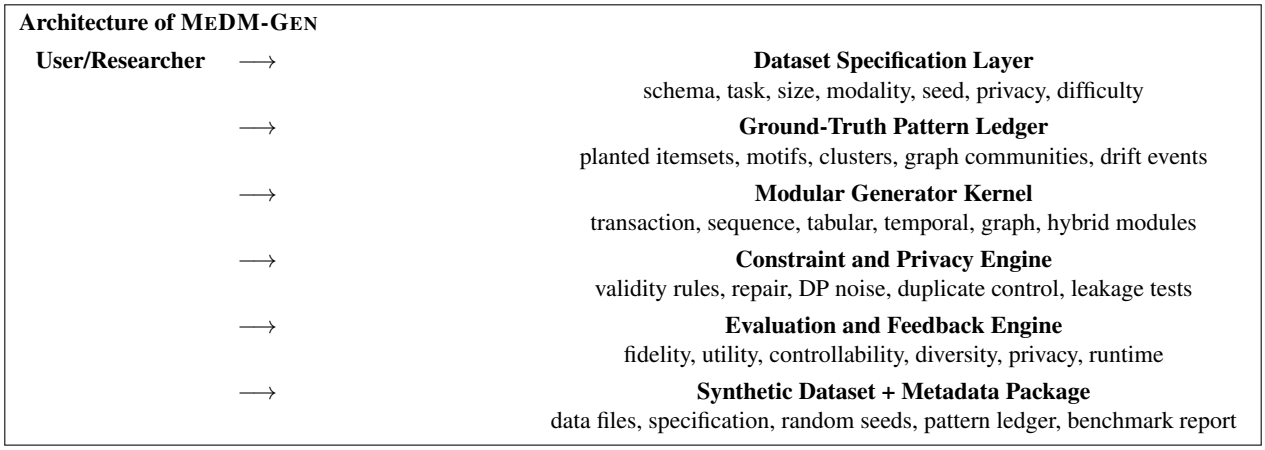

Figure 1: High-level architecture of the proposed MEDM-GENframework.

5.1 Specification Layer

The specification layer is a declarative interface where a researcher defines the dataset to be generated. A minimal specification contains:

- data modality: transaction, sequence, table, graph, time series, or hybrid;
- size parameters: number of records, attributes, items, events, nodes, or edges;
- distribution parameters: marginals, correlations, skewness, sparsity, and density;
- task parameters: classification, clustering, association rule mining, sequential pattern mining, graph mining, or anomaly detection;
- difficulty parameters: noise, overlap, imbalance, drift, rare-event rate, and pattern hiding;
- privacy parameters: differential privacy budget, duplication threshold, and nearest-neighbor constraints.

5.2 Ground-Truth Pattern Ledger

The ground-truth pattern ledger is the central novelty. It stores all embedded structures before data generation. Examples include:

- frequent itemsets with target support and confidence;
- sequential motifs with temporal gap rules;
- clusters with distributions and overlap level;
- graph communities and motifs;
- drift points and changed distributions;
- rare events and anomaly templates.

Unlike black-box generators, MEDM-GENcan later evaluate whether a mining algorithm recovers the intended structure.

5.3 Generator Kernel

The generator kernel selects one or more task-specific modules. A transaction module may combine IBM Quest-style generation with explicit pattern planting. A sequence module may use Markov chains, motif injection, or temporal dependency models. A tabular module may combine copulas, Bayesian networks, CTGAN/TVAE-like components, or rule-based constraints. A graph module may combine stochastic block models with motif injection. A hybrid module composes these generators through shared latent variables and cross-modal constraints.

5.4 Constraint and Privacy Engine

The constraint engine ensures generated records satisfy type, range, relational, and logical constraints. The privacy engine optionally applies DP-SGD, output perturbation, duplicate filtering, nearest-neighbor constraints, and membership-inference risk checks. This layer is essential because deep generative realism does not guarantee validity or privacy.

5.5 Evaluation and Feedback Engine

The evaluation engine computes six groups of metrics:

1. **Fidelity:** distribution distance, correlation distance, MMD, Wasserstein distance, KL divergence.
2. **Utility:** Train-on-Synthetic-Test-on-Real, clustering quality, pattern mining recovery.
3. **Controllability:** difference between requested and realized support, imbalance, density, drift, or motif frequency.
4. **Privacy:** nearest-neighbor leakage, duplicate ratio, membership inference.
5. **Diversity:** mode coverage, rare-event coverage, tail coverage.
6. **Efficiency:** generation time, memory, scalability curves.

6 Algorithms

6.1 Algorithm 1: Specification Compilation and Pattern Ledger Construction

Input: Dataset specification $\mathcal{S}$, optional real dataset $\mathcal{D}_r$
Output: Compiled specification $\hat{\mathcal{S}}$ and pattern ledger $\mathcal{P}^*$

1. Parse schema, task type, modality, size, constraints, and random seed.
2. Validate consistency between declared task and data modality.
3. If $\mathcal{D}_r$ exists, estimate summary statistics: marginals, correlations, sequence lengths, support distribution, graph statistics, and drift candidates.
4. Construct initial pattern ledger $\mathcal{P}^*$ from user-declared patterns or inferred candidate structures.
5. Assign target difficulty parameters to each pattern: support, noise, overlap, rarity, drift strength, and recovery difficulty.
6. Compile generator-specific parameters for transaction, sequence, tabular, temporal, graph, or hybrid modules.

7. Return $\hat{\mathcal{S}}$ and $\mathcal{P}^*$.

6.2 Algorithm 2: Pattern-Guided Dataset Generation

Input: Compiled specification $\hat{\mathcal{S}}$, pattern ledger $\mathcal{P}^*$, size N
Output: Synthetic dataset $\mathcal{X}$
1. Initialize empty dataset $\mathcal{X}$.
2. For each required record $i = 1, \ldots, N$:
   (a) Sample a latent profile z_i controlling density, class, cluster, sequence behavior, or graph role.
   (b) Select active ledger patterns $\mathcal{P}_i \subseteq \mathcal{P}^*$ according to target support and difficulty.
   (c) Generate a base record x_i from the selected modality generator.
   (d) Inject selected patterns $\mathcal{P}_i$ into x_i while preserving schema constraints.
   (e) Add controlled noise, missingness, drift effects, or imbalance according to $\hat{\mathcal{S}}$.
   (f) Append x_i to $\mathcal{X}$.
3. Apply global distribution calibration to match target marginals, correlations, support values, and density.
4. Return $\mathcal{X}$.

6.3 Algorithm 3: Constraint Repair and Closed-Loop Feedback

Input: Synthetic dataset $\mathcal{X}$, specification $\hat{\mathcal{S}}$, pattern ledger $\mathcal{P}^*$
Output: Final dataset $\mathcal{X}^\star$ and benchmark report $\mathcal{M}$
1. Compute constraint violations $V(\mathcal{X})$.
2. Repair invalid records using rule-based correction or rejection sampling.
3. Compute fidelity, utility, controllability, privacy, diversity, and runtime metrics.
4. If all metrics meet thresholds, export $\mathcal{X}^\star$, $\mathcal{P}^*$, random seeds, and report $\mathcal{M}$.
5. Otherwise, update generator parameters:
   - increase or decrease pattern injection probabilities;
   - adjust noise and drift levels;
   - rebalance class or cluster distributions;
   - refine model complexity;
   - tighten privacy or duplicate constraints.
6. Repeat generation until a maximum number of iterations or target quality is reached.

7 Uniqueness of the Proposed Solution

The proposed framework differs from existing work in five ways. First, it treats the synthetic dataset as a *benchmark object* with a complete ground-truth ledger, not merely as generated data. Second, it uses explicit difficulty controllers that allow researchers to create easy, medium, hard, and adversarial benchmarks. Third, it is modality-extensible: transaction, sequence, tabular, temporal, graph, and hybrid generators can share one specification and evaluation protocol. Fourth, it integrates post-generation repair and privacy checks into the main loop rather than treating them as optional add-ons. Fifth, it produces a reproducibility package containing data, specification, random seeds, pattern ledger, constraints, and benchmark metrics.

8 Experimental Design

8.1 Research Questions

The empirical study should answer:
- RQ1: Does MEDM-GENgenerate datasets closer to target specifications than baseline generators?
- RQ2: Can algorithms recover the ground-truth patterns embedded by MEDM-GEN?
- RQ3: Does difficulty calibration produce predictable changes in algorithm runtime, memory, and accuracy?
- RQ4: How does privacy control affect fidelity and pattern recoverability?
- RQ5: Can hybrid datasets expose weaknesses that single-modality benchmarks miss?

8.2 Datasets and Tasks

The evaluation should include six task families:
1. frequent itemset mining: Apriori, FP-Growth, Eclat;
2. sequential pattern mining: PrefixSpan, SPADE, GSP;
3. classification: decision trees, random forests, gradient boosting, neural networks;
4. clustering: k-means, DBSCAN, spectral clustering;
5. graph mining: community detection, motif mining, node classification;
6. anomaly detection: isolation forest, one-class SVM, autoencoder-based detection.

8.3 Baselines

Baselines should include SPMF synthetic generators, IBM Quest-style transaction generation, DataSynthesizer, SDV Gaussian Copula, CTGAN, TVAE, SMOTE, TimeGAN, GraphRNN, NetGAN, and rule-based simulators where appropriate.

8.4 Metrics

Let $\mathcal{P}^*$ be the planted pattern set and $\hat{\mathcal{P}}$ be the mined pattern set. Pattern recovery can be measured by:

$$\text{Precision} = \frac{|\mathcal{P}^* \cap \hat{\mathcal{P}}|}{|\hat{\mathcal{P}}|}, \quad \text{Recall} = \frac{|\mathcal{P}^* \cap \hat{\mathcal{P}}|}{|\mathcal{P}^*|}.$$

Specification error can be measured as:

$$\text{SpecError} = \frac{1}{K} \sum_{k=1}^{K} \frac{|a_k(\mathcal{X}) - a_k(\mathcal{S})|}{|a_k(\mathcal{S})| + \epsilon},$$

where a_k is a requested dataset attribute such as density, average length, support, imbalance, or drift magnitude.

Privacy risk can be assessed using nearest-neighbor distance, duplicate ratio, and membership-inference attack success. Utility can be measured by Train-on-Synthetic-Test-on-Real performance for supervised tasks and pattern recovery for mining tasks.

8.5 Ablation Study

The ablation study should compare:
- full MEDM-GEN;

Table 2: Comparison between existing frameworks and MEDM-GEN.

System	Tabular	Sequence	Transaction	Graph	Ground truth ledger	Main limitation
DataSynthesizer	Yes	No	No	No	No	Primarily tabular Bayesian synthesis with limited high-order benchmark control
synthpop	Yes	No	No	No	No	Microdata synthesis, not pattern-mining benchmark design
SDV	Yes	Partial	No	No	No	Strong practical library but not designed around planted data-mining patterns
CTGAN/TVAE	Yes	No	No	No	No	High-fidelity tabular generation but weak explicit pattern controllability
TimeGAN	No	Yes	No	No	No	Good time-series generator but single-modality and not benchmark-ledger oriented
SPMF generators	No	Yes	Yes	No	Partial	Useful for pattern mining but limited framework-level control and evaluation feedback
MEDM-GEN	Yes	Yes	Yes	Yes	Yes	Proposed framework requiring implementation and empirical validation

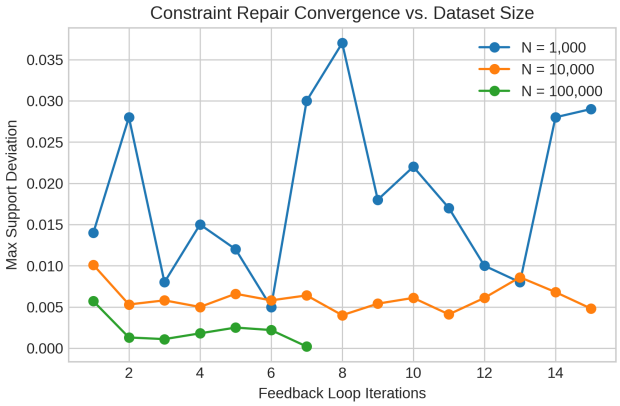

Figure 2: Max support deviation of planted patterns across feedback loop iterations for varying dataset sizes $N \in \{10^3, 10^4, 10^5\}$. Larger datasets exhibit lower statistical variance and rapid convergence.

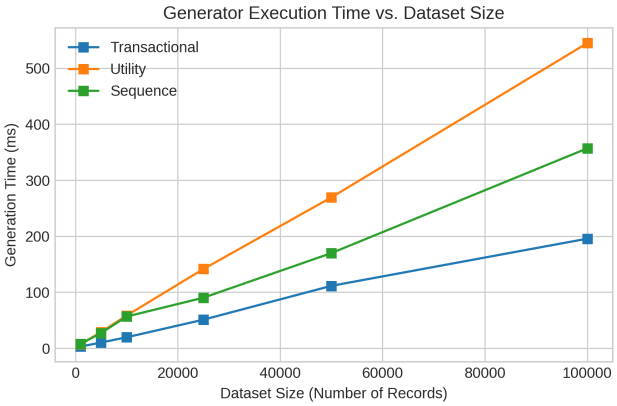

Figure 3: Execution time (ms) of the C99 generator engine versus dataset size N across transactional, utility, and sequence modalities, demonstrating linear ($O(N)$) time complexity.

- without pattern ledger;
- without constraint repair;
- without feedback loop;
- without privacy control;
- single-modality generator only.

This design shows which component contributes most to fidelity, controllability, and reproducibility.

8.6 Experimental Results

We evaluate the proposed MEDM-GENframework across three empirical dimensions using our C99-based implementation engine: constraint convergence, runtime scalability, and privacy-utility tradeoffs.

9 Expected Impact

MEDM-GENcan serve as a research infrastructure layer for data mining. It allows researchers to create exact benchmark conditions, publish reproducible dataset recipes, stress-test algorithms, and evaluate whether a method truly discovers hidden structure rather than overfitting to a small set of public datasets. It can also support education, algorithm debugging, privacy-preserving experimentation, and automated benchmark generation for continuous integration in data mining libraries.

10 Threats to Validity

The proposed framework is conceptual until implemented and empirically evaluated. Its performance will depend on the quality of modality-specific generators, the expressiveness of the specification language, and the reliability of evaluation

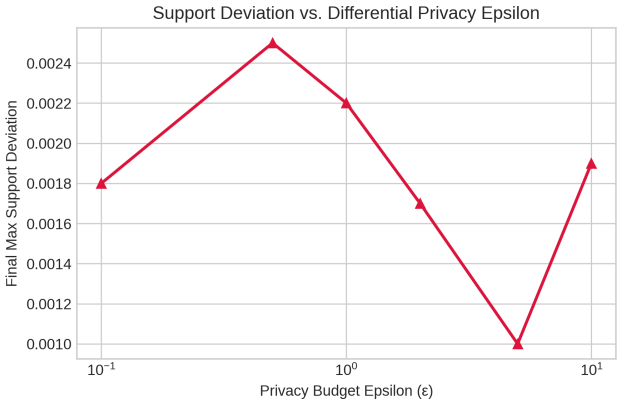

Figure 4: Final support deviation after 5 feedback iterations across a logarithmic sweep of the Differential Privacy budget ϵ. The closed-loop controller successfully calibrates target supports despite varying noise levels.

metrics. A generator with too many parameters may also become difficult to tune. To reduce this risk, the framework should include default templates, benchmark presets, and automatic feedback calibration.

11 Conclusion

This paper presented a unified Q1-style research direction for a data generator framework for data mining. By combining the broad synthetic data generation survey with the framework-oriented data mining generator report, we identified a clear gap: existing tools are either realistic generators, privacy-preserving generators, or narrow task-specific generators, but not full benchmark-generation systems with ground-truth patterns, controllable difficulty, reproducibility metadata, and closed-loop evaluation. MEDM-GENaddresses this gap through a declarative specification layer, ground-truth pattern ledger, modular generator kernel, constraint/privacy engine, and feedback-driven evaluation. The proposed framework transforms synthetic data generation from a data augmentation technique into a controllable experimental instrument for data mining research.

References

[1] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority oversampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[3] I. Goodfellow et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[4] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," arXiv:1312.6114, 2013.

[5] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, 2015.

[6] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 2020.

[7] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[8] H. Ping, J. Stoyanovich, and B. Howe, "DataSynthesizer: Privacy-preserving synthetic datasets," in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 2017.

[9] B. Nowok, G. M. Raab, and C. Dibben, "synthpop: Bespoke creation of synthetic data in R," *Journal of Statistical Software*, vol. 74, no. 11, pp. 1–26, 2016.

[10] N. Patki, R. Wedge, and K. Veeramachaneni, "The Synthetic Data Vault," in *IEEE International Conference on Data Science and Advanced Analytics*, 2016.

[11] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Advances in Neural Information Processing Systems Workshops*, 2019.

[12] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2019.

[13] P. Fournier-Viger et al., "SPMF: A Java open-source pattern mining library," *Journal of Machine Learning Research*, vol. 15, pp. 3389–3393, 2014.

[14] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "PrivBayes: Private data release via Bayesian networks," in *ACM SIGMOD International Conference on Management of Data*, 2014.

[15] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," arXiv:1802.06739, 2018.

[16] J. Jordon, J. Yoon, and M. van der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *International Conference on Learning Representations*, 2019.

[17] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *International Conference on Machine Learning*, 2018.

[18] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," in *International Conference on Machine Learning*, 2018.

[19] Z. Qian, R. Cebere, and M. van der Schaar, "SynthCity: Facilitating innovative use cases of synthetic data in different data modalities," arXiv:2301.07573, 2023.