QG Synthetic Time Series Generation: An Adapted Quantile Graph Method

Iasmyn Silva
IC/UFF, Brazil
iasmynlugon@id.uff.br

Rodrigo Chimelli Silva
IC/UFF, Brazil
rodrigochimelli@id.uff.br

Antonio A. de A. Rocha
IC/UFF, Brazil
arocha@ic.uff.br

***Abstract*—Collection and distribution of data often face regulatory and ethical barriers. Synthetic data can be a helpful solution to preserve anonymity and privacy. This study compares a new approach for generating synthetic time series based on the Quantile Graph method to different models based on statistical properties or deep-learning methods, including Ydata, a closed-source alternative. To evaluate methods, descriptive statistics and visual analysis were employed and graph measures derived from Horizontal Visibility Graphs were compared, in addition to privacy and utility metrics. Results indicate that the QG and the closed-source methods are capable of producing synthetic series that preserve many statistical and structural properties of the original data, but each exhibits limitations under specific conditions. Furthermore, the Ydata framework seems to perform better when periodicity is a concern, while the adapted QG method achieves the best privacy, when observing the tradeoff with utility.**

***Index Terms*—Synthetic Data Generation, Time Series, Quantile Method**

I. Introduction

Although many public datasets are now available, domain-specific datasets are often still needed. However, their collection and distribution frequently face regulatory and ethical barriers related to anonymity and privacy. The use of synthetic data can be a helpful solution, allowing for the development and validation of methods and models without compromising sensitive information.

Different types of data, such as tabular, images, text, or time series, are met with unique challenges. Time-series data needs to account for dependency between observations, trends, and seasonality. Ideally, a synthetic dataset should have the same statistical characteristics as its real-world counterpart, as well as preserve temporal dynamics to ensure realistic analysis.

This work proposes a simplified yet robust approach by adapting the quantile method, which transforms a time series into a complex network graph and then reconstructs it back into a time series. The adaptation consists of introducing the number of observations considered as a new parameter, which allows for the incorporation of temporal trends.

To validate the consistency of the generated data, we compared statistical measures with the original dataset. We also used the Horizontal Visibility Graph (HVG) method to examine and compare the structural properties of the resulting graphs, including node degree distribution with Jensen-Shannon divergence and Kolmogorov-Smirnov test. Furthermore, we used known statistical and deep-learning models such as Gaussian Copula to generate synthetic data as a comparative benchmark.

The remainder of the article is organized as follows: Section II reviews related works in the literature. Section III presents important concepts and information about our data. Section IV outlines the experiment, and Section V presents the results. Section VI provides the conclusions.

II. Related Work

Several methods have been proposed to generate synthetic data from time series. In [1], the authors use two different approaches to generate data. The first method substitutes the residual component in the time series. In contrast, the second method replaces the noise component with a randomized version and synthesizes seasonality using Long Short-Term Memory (LSTM) neural networks. These methods aim to preserve seasonality and trend while introducing controlled variability.

In [2], the authors explore image-based approaches, transforming time-series data into visual representations. The authors evaluate different methods, including recurrence plots, Gramian Angular Fields, and Markov transition fields, which are validated with statistical similarity metrics and machine learning tasks. Results indicate that the generated data can retain essential time-series features and can also enhance model robustness under limited data conditions.

TimeGan, a generative time-series model that extends the Generative Adversarial Network (GAN) framework is introduced in [3]. TimeGan integrates supervised and unsupervised objectives to jointly learn temporal dynamics and overall data distribution to effectively preserve temporal correlations and long-term patterns. The authors demonstrate superior performance compared to baseline methods across real-world datasets such as stock and energy related data.

Three different methods are presented in [4]. The DataFusion method aggregates data into a higher level of granularity (for example, daily to weekly). This aggregated data is used as a reference point to generate a new time series with smaller granularity (for example, by applying weighted averages of weekly values to each day). The original series is then decomposed into trend, seasonality, and residual components. Trend and seasonality are extracted and used to replace their corresponding values in the generated data, to ensure that the synthetic data retains key features of the original time

series. The ResidualReshape method, conversely, keeps the trend and seasonal components intact but replaces the residual component with newly generated noise. The DataSynthesizedGAN method adopts TimeGAN to preserve the temporal and distribution features present in the original data.

In [5], authors present MobDeep, a framework to evaluate and compare generative models for mobility data. The framework integrates both statistical (ARIMA) and Generative Adversarial Network-based (GAN) approaches to generate synthetic datasets that preserve temporal and spatial patterns of real mobility data while protecting privacy.

In [6], authors conduct a comprehensive survey on data-generating strategies applicable to Time Series Foundation Models (TSFMs) and Large Language Model-based Time Series Models (TSLLMs). The survey categorizes synthetic data generation methods into statistical, simulator-based, and data-driven approaches and reviews their integration into TSFMs and TSLLMs. It highlights how synthetic data not only improves performance but also serves as a valuable tool for probing model behavior and benchmarking. Despite these benefits, the authors emphasize current challenges, such as the limited realism of generated series.

At present, there is no consensus on how privacy and utility evaluations of synthetic data should be conducted. However, several works, among others [7],[8] and [9], propose frameworks and guidelines towards more consistent and standardized privacy and utility assessments, while papers such as [10] review commonly used privacy metrics and libraries such as SDV metrics [11] help facilitate practical implementation.

III. Preliminaries

This section introduces key concepts and the necessary background for this study, including the quantile method, Horizontal Visibility Graph, and a concise overview of the models used as a benchmark.

A. Quantile Graph Method

Given a time series $X \in \mathcal{T}$ and a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $X = \{\, x(t) \mid t \in \mathbb{N},\ x(t) \in \mathbb{R} \,\}$ and $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, the Quantile Graph (QG) method consists of a simple discretization: Q quantiles are identified and each q_i quantile is assigned to a node $n_i \in \mathcal{N}$. Two nodes n_i and n_j are then connected with a weighted arc (n_i, n_j, w_{ij}) where w_{ij} is given by the probability that a point in quantile q_i is followed by a point in quantile q_j. Repeated transitions between quantiles result in larger weights. This is the forward mapping (Algorithm 1) part of the process, which results in an adjacency matrix representing connections between nodes.

The inverse map (Algorithm 2) consists of normalizing the weighted adjacency matrix W, which then becomes a Markov transition matrix. The time series is reconstructed by moving from node n_i to n_j with probability w_{ij} and choosing a random value between the limits of the corresponding quantile q_j.

It is a numerically simple method with only one free parameter Q, the number of quantiles (nodes), which is typically much smaller than T, resulting in networks that are weighted, directed, and may contain self-loops ([12],[13],[14]). Figure 1 illustrates a simple example with a toy time series consisting of 20 random observations (22, 18, 2, 24, 15, 37, 7, 39, 27, 5, 34, 10, 6, 9, 45, 24, 29, 7, 23, 24) split into four quantiles with lower and upper limits {[2,8],(8, 22],(22, 27],(27, 45.0]}.

Algorithm 1 Quantile Method: Forward Map

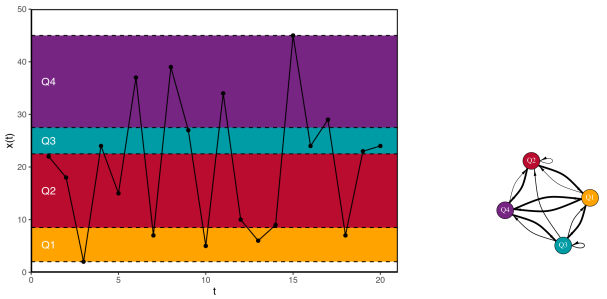
Require: X original timeseries of length T
Require: Q number of quantiles
Ensure: W weighted adjacency matrix $q \times q$
Ensure: L set of tuples of length Q

$W \leftarrow Q \times Q$ matrix with all values set to 0
$L \leftarrow$ set of tuples of length Q
$S \leftarrow$ empty sequence of length T
$Q^* \leftarrow$ vector of quantiles
for $k = 1$ to T **do** ▷ *divide X into Q quantiles*
  assign x_k to its corresponding quantile q_j^*, $j \leq Q$
  $S_k \leftarrow q_j^*$ ▷ *we now have a sequence of quantiles*
for $i = 1$ to Q **do**
  $L_i \leftarrow (\min(q_i^*), \max(q_i^*))$
for row r in W **do**
  for column c in W **do**
    for $k = 1$ to $(T - 1)$ **do**
      $w \leftarrow W(r, c)$
      if $s_k = q_r^*$ and $s_{(k+1)} = q_c^*$ **then**
        $w \leftarrow w + 1$
        $W(r, c) \leftarrow w$
return W, L

Algorithm 2 Quantile Method: Inverse Map

Require: W weighted adjacency matrix
Require: L set of tuples with lower and upper bounds of each quantile q
Require: T desired length of synthetic timeseries
Ensure: Y synthetic timeseries of length T

$Y \leftarrow$ empty sequence of length T
$q \leftarrow$ first quantile, chosen
$l \leftarrow L_q$
$v \leftarrow$ value between (l_1, l_2) ▷ *chosen randomly from all values between lower and upper bounds of quantile q*
$Y_1 \leftarrow v$
for $y = 2$ to T **do**
  $V \leftarrow W_{(q,*)}$ ▷ *row q of matrix A*
  normalize V ▷ *vector of transition probabilities with all values between 0 and 1*
  $next\_q \leftarrow$ choose column from $W_{(q,*)}$ ▷ *chosen according to transition probability in normalized V*
  $l \leftarrow L_{next\_q}$
  $v \leftarrow$ value between (l_1, l_2)
  $Y_y \leftarrow v$
  $q \leftarrow next\_q$
return Y

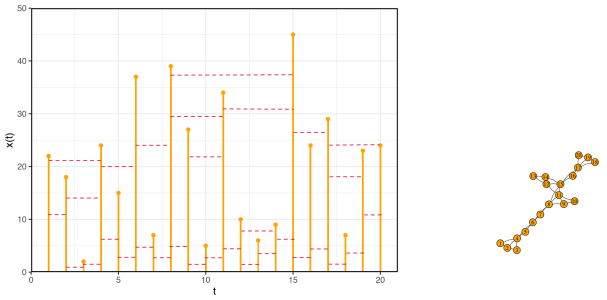


(a) Toy Timeseries (b) QG Network

Fig. 1: QG Method

B. Horizontal Visibility Graph

The Horizontal Visibility Graph (HVG) is a refinement of the Visibility Graph algorithm. Both map a time series into a graph by first assigning each observation to a vertical bar of height equal to the numerical value of the observation, and each node corresponds to one observation ([14], [15]). Then two nodes are connected if a horizontal line that joins x_i and x_j can be drawn without intersecting any intermediate data height, i.e., $x_i, x_j > x_n, \forall n \in (i, j)$. Figure 2 illustrates an example with the same toy dataset.



(a) Toy Timeseries (b) HVG Network

Fig. 2: HVG Method

C. Benchmark Models

To evaluate the quality of our synthetic data, we employed a set of benchmark models based on statistical, deep-learning and GAN-based approaches. Each model was used to generate synthetic datasets under the same experimental conditions, allowing for a systematic comparison with the results obtained from our proposed method.

- Gaussian Copula: A statistical model that estimates the joint probability distribution of a dataset using copulas by modeling each marginal distribution independently, transforming them into a Gaussian space, and capturing dependencies through a covariance matrix
- Tabular Variational Autoencoder (TVAE): A deep learning model based on variational autoencoders. It encodes the real data into a latent space and decodes synthetic samples by learning continuous latent representations, which helps preserve complex relationships and non-linear dependencies among variables.
- Probabilistic AutoRegressive Model (PAR): A probabilistic sequential model that generates each record conditioned on previously generated ones, capturing temporal or ordered dependencies.
- Conditional Tabular GAN (CTGan): A GAN-based model. It uses a conditional generator and training-by-sampling strategy to handle data imbalance and mode collapse, effectively learning diverse and realistic distributions.
- Ydata Fabric: a closed-source alternative advertised as the best generative AI model for synthetic time-series data generation [16]

Gaussian, TVAE, PAR and CTGan data were all generated with the Synthetic Data Vault Python library [11]. Ydata Fabric data was generated with the ydata Python library [17].

IV. Method

This study proposes a new approach to the QG method, considering two or more observations jointly. While the original method assigns each q_i quantile to a n_i node, the proposed method first groups two or more consecutive observations into a state $s = q_i ... q_n, n \in \mathbb{N}$ and then each state to a node n_i^* (Algorithm 3). The total number of possible states is Q^O, since it is likely that a value in q_i is followed by another value in the same quantile. By introducing the number of observations $o \in \mathbb{N}, o < t$ as a new parameter, the aim is to capture temporal trends rather than focusing on a fixed point in time.

The inverse map (Algorithm 4) of the proposed algorithm needs to be adjusted to consider the probability of transitioning from one state to another, instead of the transition between quantiles. When a state is defined, its corresponding value is selected from the limits of its last quantile.

Table I presents the original quantiles and nodes along with their corresponding states, considering the same toy time series from previous examples and two observations to form a state.

Notice that in the example, the number of nodes in the network when using the adapted method (12) is higher than the number of nodes when using the original QG method (4), but lower than the total number of possible states (16). This occurs because not all possible states are present in the given toy data, which would result in an adjacency matrix with entries equal to zero. The final number of states is also influenced by the smoothness of the time series. When a time series exhibits incremental changes, the resulting number of states will be lower compared to a series with abrupt fluctuations.

A. Experiment

The real data set used in this work consists of a collection of data from 21 different cellphone towers in the state of Rio de Janeiro, Brazil. Data were collected over a one-week period in 2021, with each observation corresponding to the number of devices connected to the network at five-minute intervals, yielding a total of 2016 observations per time series. Data were obtained through a partnership with a Brazilian telecoms

Algorithm 3 Adapted Quantile Method: Forward Map

Require: X original timeseries of length T
Require: Q number of quantiles
Require: O number of observations to form a state
Ensure: W weighted adjacency
Ensure: L set of tuples

$Z \leftarrow Q^O$
$W \leftarrow Z \times Z$ matrix with all values set to 0
$L \leftarrow$ set of tuples of length Q
$S \leftarrow$ empty sequence of length T
$Q^* \leftarrow$ vector of quantiles
$S^* \leftarrow$ vector of all possible states ▷ *permutation of quantiles, including repetitions*
for $k = 2$ to T **do** ▷ *divide X into Q states*
    assign x_{k-1} to its corresponding quantile q_i^*, $i \leq Q$
    assign x_k to its corresponding quantile q_j^*, $j \leq Q$
    $s \leftarrow q_i^* q_j^*$ ▷ *combine both quantiles to form a state*
    $S_k \leftarrow s$ ▷ *we now have a sequence of states*
for $i = 1$ to Q **do**
    $L_i \leftarrow (\min(q_i^*), \max(q_i^*))$
for row r in W **do**
    for column c in W **do**
        for $k = 2$ to (T) **do**
            $w \leftarrow W(r, c)$
            if $s_k = s_r^*$ and $s_{(k+1)} = s_c^*$ **then**
                $w \leftarrow w + 1$
                $W(r, c) \leftarrow w$
return W, L

Algorithm 4 Adapted Quantile Method: Inverse Map

Require: W weighted adjacency matrix
Require: L set of tuples with lower and upper bounds of each quantile q
Require: T desired length of synthetic timeseries
Ensure: Y synthetic timeseries of length T

$Y \leftarrow$ empty sequence of length T
$s \leftarrow$ first state, chosen randomly
$q \leftarrow$ last quantile of s ▷ *if we have a state $q_1 q_2$ then $q \leftarrow q_2$*
$l \leftarrow L_q$
$v \leftarrow$ value between (l_1, l_2) ▷ *chosen from all values between lower and upper bounds of quantile q*
$Y_1 \leftarrow v$
for $y = 2$ to T **do**
    $V \leftarrow W_{(s,*)}$ ▷ *row s of matrix A*
    normalize V ▷ *vector of transition probabilities with values between 0 and 1*
    $s \leftarrow$ choose column from $W_{(s,*)}$ ▷ *chosen according to transition probability in normalized V*
    $q \leftarrow$ last quantile of s
    $l \leftarrow L_{next\_q}$
    $v \leftarrow$ value between (l_1, l_2)
    $Y_y \leftarrow v$
return Y

TABLE I: Quantiles to States

t	x(t)	q	n	s	n^*
1	22	q_2	n_2	-	-
2	18	q_2	n_2	q_2q_2	n_5^*
3	2	q_1	n_1	q_2q_1	n_4^*
4	24	q_3	n_3	q_1q_3	n_2^*
5	15	q_2	n_2	q_3q_2	n_8^*
6	37	q_4	n_4	q_2q_4	n_6^*
7	7	q_1	n_1	q_4q_1	n_{10}^*
8	39	q_4	n_4	q_1q_4	n_3^*
9	27	q_3	n_3	q_4q_3	n_{12}^*
10	5	q_1	n_1	q_3q_1	n_7^*
11	34	q_4	n_4	q_1q_4	n_3^*
12	10	q_2	n_2	q_4q_2	n_{11}^*
13	6	q_1	n_1	q_2q_1	n_4^*
14	9	q_2	n_2	q_1q_2	n_1^*
15	45	q_4	n_4	q_2q_4	n_6^*
16	24	q_3	n_3	q_4q_3	n_{12}^*
17	29	q_4	n_4	q_3q_4	n_9^*
18	7	q_1	n_1	q_4q_1	n_{10}^*
19	23	q_3	n_3	q_1q_3	n_2^*
20	24	q_3	n_3	q_3q_3	n_2^*

company. A description of the original dataset is presented in Table II. In this study, an unique identifier (ID) was associated for each tower, based on its geographic location and the data were aggregated by tower and timestamp, resulting in a processed dataset with three variables: (i) Timestamp, (ii) Tower ID, and (iii) Number of connected devices.

TABLE II: Dataset description

Variable	Description
Timestamp	Date and time
Area code	Local area code for the phone number connected to the phone tower
Country code	Country code for the phone number connected to the phone tower
Latitude	Geographic latitude of the tower
Longitude	Geographic longitude of the phone tower
N	Number of connected devices

The number of quantiles was defined as $Q = 2T^{1/3}$ [13]. The number of observations in a state was variable, and experiments were conducted with 2, 3, and 5 observations.

V. Results

The evaluation of results was conducted by analyzing nine distinct time series for each cellphone tower: the original series, the three synthetic series produced using the adapted QG method and the synthetic series generated by Ydata, PAR, TVAE, CTGAN and Gaussian Copule models.

A. Quantitative Analysis

First, a simple quantitative analysis was performed, comparing the average, standard deviation, minimum and maximum values of the synthetic data generated with the original one. For conciseness purposes, results for 3 of the 21 towers are summarized in Table III, but the results were similar for all.

It can be observed that the QG method exhibits statistics more similar to the original, whereas Ydata generates data differ the most. For Ydata-generated data, this is especially evident in the maximum values, which are sometimes more than twice the original (towers 2 and 726) and in other cases less than half (tower 732). Both Gaussian and CTGAN

TABLE III: Descriptive Statistics

Tower	Data	Min	Average	SD	Max
2	Original	0	34	22	145
2	Ydata	0	116	64	307
2	Gaussian	1	34	22	124
2	CTGan	0	42	25	130
2	TVAE	0	26	16	84
2	PAR	0	34	16	90
2	QG (2 obs)	0	36	27	143
2	QG (3 obs)	0	27	6	65
2	QG (5 obs)	0	33	25	145
726	Original	0	86	83	326
726	Ydata	0	353	351	1236
726	Gaussian	0	93	89	326
726	CTGan	0	80	77	326
726	TVAE	0	81	82	326
726	PAR	0	89	49	264
726	QG (2 obs)	0	108	87	326
726	QG (3 obs)	0	103	79	326
726	QG (5 obs)	0	80	80	325
732	Original	0	519	275	1159
732	Ydata	0	299	127	557
732	Gaussian	0	522	272	1149
732	CTGan	0	481	306	1159
732	TVAE	0	559	279	1119
732	PAR	0	512	169	1131
732	QG (2 obs)	0	335	187	1139
732	QG (3 obs)	0	407	202	1149
732	QG (5 obs)	0	569	273	1159

methods also have similar metrics to the original, for all cases. TVAE-generated data are a close match to the original for towers 726 and 732, but underestimate all values for tower 2. Data generated with the PAR method have a lower SD and maximum values for all cases, which indicates a dampened variance. For QG-generated data, the average and SD diverge the most when using only 2 or 3 observations, compared to when using 5. This most likely happens because tower 732 has a broader range of values (Figure 3), so temporal trends would be better observed with more observations.

B. Visual Analysis

In addition to descriptive statistics, we used visual analysis to assess the similarity between the generated and original series. Figure 3 presents the behavior of the same three towers as before under the different methods. It is clear that, while the QG method better captures the range of the original data, Ydata generally generates data that more closely resembles its periodicity. Meanwhile, Gaussian, CTGAN, TVAE and PAR data are flatter, with too much noise and too little cyclical variation. Among the QG variations, using only two or three observations produces less stable trajectories and, in some cases (such as using three observations for tower 2), fails to capture fluctuations. These differences suggest that the QG method's performance is sensitive to the number of observations considered, while Ydata consistently emphasizes periodicity and all others fail to replicate the underlying temporal dependencies present in the real series.

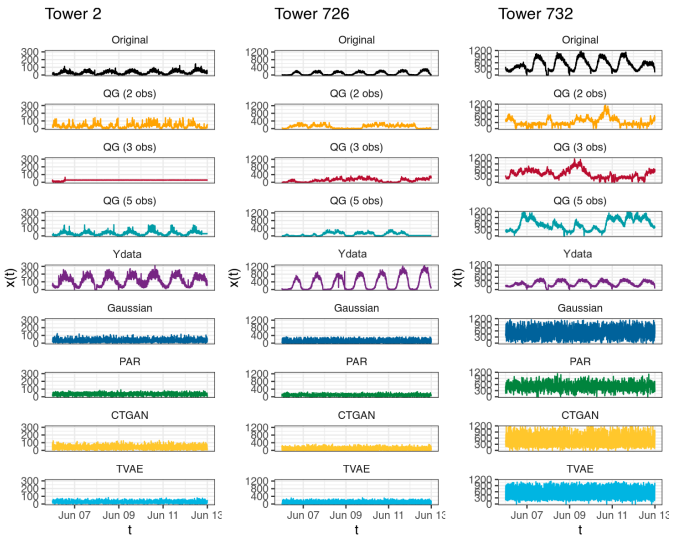


Fig. 3: Generated x Original Data

C. HVG

Next, we applied the HVG method to generate complex network graphs from the data. Each observation is assigned to a node and connections between nodes are established according to the horizontal visibility criterion. Since visualizing differences and similarities is challenging when the graph contains many nodes and edges, we computed graph theory measures, such as diameter, distance and clustering coefficient (CC). The absolute difference between original data graphs and generated data graphs can be found in Table IV. Additionally, average betweenness and closeness centrality were computed; however, since their values were consistently close to zero across all graphs, for conciseness, were omitted from the table. Figure 4 illustrates the resulting graphs for tower 726.

Overall, the clustering coefficient remained similar across all cases. Diameter and average distance, however, revealed a few notable exceptions. The Gaussian, CTGAN, TVAE, and PAR methods tended to produce graphs with systematically smaller diameters and distances for all towers, indicating a compression of network structure. The QG method with three observations for Tower 2 showed the most evident difference, consistent with the nearly linear behavior visible in Figure 3. Another significant case was observed for the Ydata method in Tower 732, which diverges from all QG configurations, exhibiting a negative difference from the original in both distance and diameter, which indicates a compression of the original structure not present in the QG generated data.

Vertex degree distributions were also compared using the Jensen–Shannon divergence (JSD) and the Kolmogorov–Smirnov (KS) test. Results relative to the graphs generated from the original data can be found in Table V.

With the KS test, we do not reject the null that the data compared come from the same distributions for most cases, with the exception of the QG method with 3 observations for

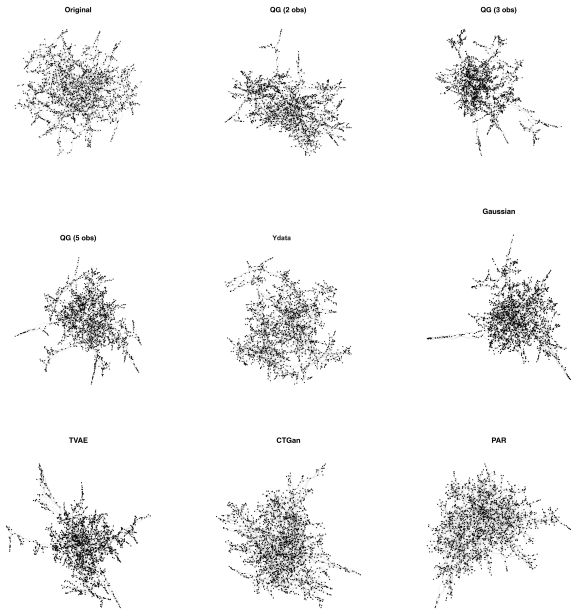


Fig. 4: HVG Generated Graphs for Tower 726

TABLE IV: HVG Statistics

Tower	Data	Diameter	Distance	CC
2	Original	32	12	0.319
2	Ydata	+6	+3	+0.008
2	Gaussian	-14	-4	+0,011
2	CTGan	-12	-3	+0,005
2	TVAE	-13	-3	+0,011
2	PAR	-10	-3	+0,011
2	QG (2 obs)	-3	-0	+0.006
2	QG (3 obs)	+462	+165	-0.047
2	QG (5 obs)	+10	+1	-0.008
726	Original	+64	21	0.326
726	Ydata	+18	+7	-0.026
726	Gaussian	-46	-13	0,005
726	CTGan	-45	-12	0,009
726	TVAE	-42	-12	0,006
726	PAR	-46	-13	0,008
726	QG (2 obs)	-12	-2	+0.015
726	QG (3 obs)	-20	-9	+0.034
726	QG (5 obs)	+15	+0.56	+0.027
732	Original	63	26	0.346
732	Ydata	-13	-6	-0.013
732	Gaussian	-45	-18	-0,012
732	CTGan	-41	-17	-0,016
732	TVAE	-39	-17	-0,011
732	PAR	-31	-13	-0,004
732	QG (2 obs)	+4	+6	+0.015
732	QG (3 obs)	+4	+3	+0.012
732	QG (5 obs)	+3	+0.06	+0.013

Tower 2, confirming its unstable behavior previously observed in Figure 3. JSD shows a more erratic behavior, with some cases it indicating that the distributions are similar, such QG with 3 and 5 observations and Gaussian for tower 732, QG with 2 and 5 observations and Gaussian for Towers 2 and PAR and QG with 2 observations for tower 726, while others exhibit evidences of data drift, such as Ydata for tower 726. This indicates that similarity in temporal patterns does not necessarily translate into structural similarity in their corresponding visibility graphs.

TABLE V: HVG Statistics

Tower	Data	KS (p-value)	JSD
2	Ydata	0.866	0.296
2	Gaussian	0.418	0.002
2	CTGan	0.396	0.003
2	TVAE	0.866	0.172
2	PAR	0.749	0.286
2	QG (2 obs)	0.937	0.003
2	QG (3 obs)	0	0.528
2	QG (5 obs)	0.99	0.002
726	Ydata	0.374	0.999
726	Gaussian	0.279	0.443
726	CTGan	0.142	0.415
726	TVAE	0.374	0.179
726	PAR	0.465	0.003
726	QG (2 obs)	0.565	0.003
726	QG (3 obs)	0.985	0.286
726	QG (5 obs)	0.999	0.181
732	Ydata	0.153	0.41
732	Gaussian	0.985	0.003
732	CTGan	0.990	0.170
732	TVAE	0.996	0.169
732	PAR	0.998	0.275
732	QG (2 obs)	0.999	0.401
732	QG (3 obs)	0.994	0.002
732	QG (5 obs)	0.99	0.002

D. Privacy and Utility Evaluation

In order to compare the synthetic generators with regard to privacy, four metrics were chosen: MIA (Membership Inference Attack) [9], Disclosure Protection and New Row Synthesis, and DCR (Distance to Closest Record) [11]. The MIA and Disclosure Protection metrics respectively assess the protection against membership and attribute inference attacks, while DCR is based on the distance to the closest record between synthetic data points and a subset of sampled real data, and New Row Synthesis calculates the percentage of synthetic rows that are not in the real dataset. To assess utility the task of predicting number of connections was chosen with the paradigm of TSTR (Train on Synthetic Test on Real) [9] resulting in the MAE, RMSE and R2 metrics. When interpreting the results, the goal for each metric varies. For DCR, Disclosure Protection, New Row Synthesis, and R^2, a higher value indicates better performance. Conversely, for RMSE and MAE, a lower value is better. The MIA metric is unique, where a score closer to 0.5 is ideal, as this signifies that an attacker's ability to infer membership is no better than a random guess.

As shown in Table VI, Ydata is the clear winner in terms of privacy, achieving the highest Disclosure Protection score of 0.59. However, the other privacy metrics are too similar across all generators to offer any further differentiation, making Disclosure Protection the key indicator.

TABLE VI: Synthetic Data Evaluation Metrics

Generator	Disclosure Protection	DCR	MIA	New Row Synthesis	R^2	MAE	RMSE
Real Data	0.000	0.042	0.504	0.301	0.385	127.759	270.656
Ydata	0.596	0.051	0.498	0.986	0.014	245.733	342.804
GAN	0.456	0.049	0.506	0.991	0.140	198.865	320.164
QG (2 obs)	0.401	0.047	0.498	0.992	0.139	199.065	320.296
QG (3 obs)	0.345	0.046	0.499	0.992	0.052	206.861	336.158
QG (5 obs)	0.378	0.049	0.508	0.992	-0.167	233.246	372.935
TVAE	0.414	0.048	0.502	0.990	0.114	201.695	324.969
PAR	0.329	0.047	0.497	0.993	0.146	194.154	318.992
Gaussian Copula	0.422	0.048	0.498	0.992	0.114	200.342	325.005

This apparent privacy advantage is misleading. Excluding the failed QG (5 obs) model (which had a negative R^2), Ydata performed the worst on every single utility metric. This strongly suggests that its high privacy score stems not from sophisticated data protection, but from a fundamental failure to learn and replicate significant patterns from the original data.

Conversely, the PAR model achieved the highest utility but at the cost of lower privacy, highlighting the classic trade-off. The best balance is found in models like PS2, which nearly matches PAR's utility while offering a 25% higher Disclosure Protection score. Therefore, QG (2 obs), TVAE, and Ydata emerge as the top contenders for effectively balancing the dual objectives of data utility and privacy.

VI. Conclusion

Despite the growing availability of public datasets, many applications still require tailored data, but regulatory and ethical concerns can restrict collection and distribution. Synthetic data offers a practical solution that preserves privacy and anonymity. In this work, we explored and compared two different approaches for generating synthetic time series: the Ydata framework, deep-learning methods such as Gaussian Copula, TVAE, and PAR and an adapted version of the Quantile Graph (QG) method.

The results indicate that the QG and Ydata methods are capable of producing synthetic time series that resemble real-world data, but each exhibits limitations under specific conditions. Overall, the Ydata framework seems to perform better when periodicity is a concern, while the adapted QG method better captures the range of the original data. Meanwhile, Gaussian Copula, TVAE, CTGan and PAR methods show less promising results, failing to capture both temporal patterns and ranges. Furthermore, while Ydata achieved the best privacy, when observing the tradeoff with utility the QG method showed the best results. Future research could focus on refining strategies for parameter selection (number of observations) and on deciding which methods are the most appropriate for assessing similarity between time series, and also expand the study to different domains to evaluate the generalizability and robustness of the proposed approach.

References

[1] I. I. Ungureanu, R.-L. Portase, M. Dinsoreanu, C. Lemnaru, and R. Potolea, "Hybrid statistical and ai methods for synthetic time series generation in practical applications," in *2024 IEEE 20th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2024.

[2] S. Oh, S. Oh, J. Lee, D. Kim, M. Hahn, and J. Kim, "Exploring image-based approaches for effective synthetic time-series data generation," in *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2025.

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

[4] R.-L. Portase, A.-M. Dragotoniu, C. Lemnaru, M. Dinsoreanu, and R. Potolea, "Advancing iot data utilization: Generating and evaluating synthetic time series data," in *2024 IEEE 20th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2024.

[5] I. F. Ribeiro, G. Comarela, A. A. A. Rocha, and V. F. S. Mota, "Towards a framework to evaluate generative time series models for mobility data features," *Journal of Internet Services and Applications*, 2024.

[6] X. Liu, T. Aksu, J. Liu, Q. Wen, Y. Liang, C. Xiong, S. Savarese, D. Sahoo, J. Li, and C. Liu, "Empowering time series analysis with synthetic data: A survey and outlook in the era of foundation models," 2025.

[7] M. Hernandez, P. A. Osorio-Marulanda, M. Catalina, L. Loinaz, G. Epelde, and N. Aginako, "Comprehensive evaluation framework for synthetic tabular data in health: fidelity, utility and privacy analysis of generative models with and without privacy guarantees," *Frontiers in Digital Health*, vol. 7, p. 1576290, 2025.

[8] A. D. Lautrup, T. Hyrup, A. Zimek, and P. Schneider-Kamp, "Syntheval: a framework for detailed utility and privacy evaluation of tabular synthetic data," *Data Mining and Knowledge Discovery*, vol. 39, no. 1, p. 6, 2025.

[9] G. Santangelo, G. Nicora, R. Bellazzi, and A. Dagliati, "How good is your synthetic data? synthro, a dashboard to evaluate and benchmark synthetic tabular data," *BMC Medical Informatics and Decision Making*, vol. 25, no. 1, p. 89, 2025.

[10] P. A. Osorio-Marulanda, G. Epelde, M. Hernandez, I. Isasa, N. M. Reyes, and A. B. Iraola, "Privacy mechanisms and evaluation metrics for synthetic data generation: A systematic review," *IEEE Access*, vol. 12, pp. 88 048–88 074, 2024.

[11] *Synthetic Data Metrics*, DataCebo, Inc., 05 2025, version 0.21.0. [Online]. Available: https://docs.sdv.dev/sdmetrics/

[12] A. S. L. O. Campanharo, M. I. Sirer, R. D. Malmgren, F. M. Ramos, and L. A. N. Amaral, "Duality between time series and networks," *PLOS ONE*, 2011.

[13] A. Campanharo, E. Doescher, and F. M. Ramos, "Application of quantile graphs to the automated analysis of eeg signals," *Neural Processing Letters*, 2020.

[14] V. F. Silva, M. E. Silva, P. Ribeiro, and F. Silva, "Time series analysis via network science: Concepts and algorithms," *WIREs Data Mining and Knowledge Discovery*, 2021.

[15] B. Luque, L. Lacasa, F. Ballesteros, and J. Luque, "Horizontal visibility graphs: Exact results for random time series," *Phys. Rev. E*, 2009.

[16] YData. (2023) The best generative ai model for time-series synthetic data generation. https://ydata.ai/resources/the-best-generative-ai-model-for-time-series-synthetic-data-generation. Accessed: 2025-10-26.

[17] Y. Team, "Ydata sdk," https://docs.sdk.ydata.ai/latest/, version 3.0.8, accessed June, 2025.