SYNTHETIC DATA APPLICATIONS FOR FEDERAL STATISTICS

Taylor J. Wilson, Alan Weisel, Ellie Mamantov
Taylor.Wilson@revealgc.com, Alan.Weisel@revealgc.com, Ellie.Mamantov@revealgc.com

Reveal
GLOBAL CONSULTING

Abstract

The systematic creation of data simulated for testing purposes (i.e., synthetic data) and its intersection with data security concerns that are present in most federal agencies remains a key challenge. This paper explores the diverse ways in which synthetic data are used in a federal context, the unique concerns federal agencies face when integrating it into an existing workflow and presents solutions that have worked to deploy synthetic data systems. Some of those concerns are centered around data security protections like Title XIII and Title XXVI which make the use of certain data highly sensitive in the development and testing of new processes. Another need is for synthetic data to test systems that are built for brand new surveys or collection initiatives for which no data yet exists. Synthetic data can eliminate disclosure risk and ensure that tests involving new, yet-to-be-collected data are robust. The paper also explores different ways of generating synthetic data and discusses why certain approaches are well-suited to the federal context.

I. Introduction

Synthetic data encompasses a range of definitions within the federal context. For this research, the computer science perspective is adopted where synthetic data is simulated for testing and development. This contrasts with the definition from survey methodology, where synthetic data may involve combining multiple data sources to produce reliable estimates for specific and granular populations. Such synthesis is particularly useful when a single data source fails to provide sufficient insight on its own.[1]

Federal agencies often turn to synthetic data when real, historical datasets are either unavailable or cannot be used due to security and privacy concerns. This necessity is highlighted in scenarios where agencies are tasked with developing and testing systems intended for new surveys or data collection initiatives that have no precedent. The utilization of synthetic data not only addresses these gaps but also plays a crucial role in ensuring that new systems are both robust and secure before their actual implementation. Because of the diverse methods of generating these data, it is often not clear to federal stakeholders how much effort is required to achieve their goals. This paper proposes a framework for determining what complexity of synthetic data are required for any use case and suggests which methods are best suited for matching that desired complexity.

II. Defining the Problem

Statistical agencies are frequently tasked with handling data safeguarded by U.S. legislative titles XIII and XXVI. Title XIII entrusts agencies like the Census Bureau with the collection of personal information from individuals and businesses while mandating strict confidentiality; it ensures that personal identifiers are not disclosed, utilized adversely, or repurposed outside of statistical analysis.[2] Similarly, Title XXVI allows the Internal Revenue Service to disseminate tax return information for statistical uses in government censuses and economic measures, mandating rigorous protection protocols, including audits and controlled access.[3]

[1] https://www.census.gov/about/what/synthetic-data.html
[2] https://www.census.gov/history/www/reference/privacy_confidentiality/title_13_us_code.html
[3] https://www.census.gov/history/www/reference/privacy_confidentiality/title_26_us_code_1.html

The enactment of these titles is crucial for preserving American citizens' privacy, maintaining public trust, and enhancing the data's integrity. The benefits of these title protections do come at an operational cost in that they can also impede the development and testing of new data systems. When building such systems, the actual data must typically be processed to identify any security weaknesses. However, using sensitive data for such tests could inadvertently expose it to risks. This paradox underscores the value of synthetic data, which simulates the characteristics of real data without containing sensitive information, allowing for thorough testing without the risk of compromising privacy.

The development of new processing systems often involves many professionals who, under the principles of a zero-trust framework, should only access information on a need-to-know basis.[4] Synthetic data mitigates the risk of exposure, as fewer individuals need to handle actual sensitive data during the development phase.

The challenge also extends to testing procedures for new data collection efforts. An example is the Annual Integrated Economic Survey launched in March 2024, consolidating seven industry-specific surveys into one.[5] It not only recycled some questions but also introduced new ones. Without prior data on these new items, the testing of processing systems could not wait for actual data collection. In this use case, synthetic data was instrumental, allowing the systems to be tested and refined, ensuring that when real data are collected, it is processed accurately and securely. This preemptive testing was essential in validating the systems' capabilities to handle data through all stages—from collection to dissemination—effectively ensuring the success of the survey before its official deployment.

The different stages of the data life cycle in this survey required synthetic data in different forms to test, develop, and deploy. In some cases, the data did not need to be very accurate but rather just a structural emulation of the real data, while other users of the data required more accurate simulated information to build out and test their processes. Because of this, a framework became necessary to determine what kind of synthetic data were necessary to meet the needs of the various stakeholders.

Framework for Synthetic Data Complexity

Synthetic data serves as a powerful tool across many applications, its utility dictated by the complexity of its structure and its resemblance to real data. To effectively leverage synthetic data, understanding its inherent complexity is essential. The complexity of synthetic data can be mapped along a spectrum, from low to high, based on the intricacy of its features and the depth of its similarity to actual datasets. This framework categorizes synthetic data into three primary levels of complexity:

Low Complexity Synthetic Data

- **Data Type Matching**: At the most basic level, synthetic data should match the data types of the actual data. This includes simple attributes such as integers, floats, strings, dates, etc.
- **Variable Name Matching**: The synthetic dataset features variable names that correspond to those in the real dataset, ensuring that the system can recognize and process the synthetic data in a manner consistent with actual data.

[4] https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/
[5] https://www.census.gov/programs-surveys/aies/about.html

Medium Complexity Synthetic Data

- **Preservation of Numeric Characteristics**: This tier of synthetic data preserves basic statistical properties such as means, standard deviations, ranges, and other moments of the distributions of the original data.
- **ID Structures Preservation**: Unique identifiers and their structures, which might include specific formatting or coding schemes, are accurately replicated in the synthetic data.
- **Defined and Implemented Relationships**: Beyond individual variable characteristics, this complexity level requires the establishment and maintenance of relationships between variables, reflecting dependencies present in the real data.

High Complexity Synthetic Data

- **Indistinguishable from Reality**: The pinnacle of synthetic data complexity is achieved when the data are virtually indistinguishable from the real thing. This includes all the characteristics of the lower levels, with additional nuances such as the replication of complex patterns, trends, anomalies, and the intricate, multi-layered relationships that are inherent in the actual data.

By defining requirements within this framework, one can more accurately determine the kind of synthetic data needed for a specific use case. Lower complexity synthetic data may suffice for basic system testing and development where the focus is on format and field-level validation. As the use case becomes more nuanced, requiring statistical analysis or the testing of data processing algorithms, medium complexity data becomes necessary. Finally, for applications such as machine learning model training, where the model's performance is tightly coupled with the data's idiosyncrasies, or disclosure protection practices, then high complexity synthetic data are essential.

III. Generation Methods

The selection of a synthetic data generation method is pivotal and depends largely on the intended application's complexity and the criticality of data accuracy. This section outlines and compares two principal methods: rules-based generation and model-based generation, providing insights into their applicability.

3.1 Rules-Based Generation

In rules-based generation, every parameter is defined before data production. These parameters are used to make the rules that ultimately produce the data. For example, a variable could have a set of parameters associated with it like a character data type, a length of two, four possible values (e.g., NY, VA, MD, TX) distributed evenly across the data. Parameters can also be defined relationally so that one variable can inform another. For example, if a record is NY in the character variable, then another variable may take a random numeric value between 1 and 10. This approach yields data in a deterministic fashion, providing precise control over the output. Such predictability is particularly beneficial for security testing, where it is important to understand how each record was produced. For process testing too, where introducing known errors to simulate real-world data issues is necessary, rules-based generation excels. It allows for the deliberate introduction of specific errors, enabling the construction of highly controlled test scenarios.

Consider an outlier detection method developed by a federal agency that attempts to locate anomalous records. In economic data collection, there is a persistent problem where respondents will provide data in the incorrect units asked by the survey question. For example, the Annual Integrated Economic Survey in the U.S. Census Bureau collects business revenue in 'thousands of dollars.' This means that the respondent is expected to provide a revenue of $100,000 as $100. If the instruction is misunderstood and the respondent enters their whole dollar amount, then the system will read their business revenue as $100,000,000—a factor of 1000 too large. Because this is a predictable issue, data processing systems need to devise a method of catching potential issues like this and correcting them. The role of synthetic data is to produce a specific number of these errors throughout the dataset so that such a system can be tested against mock survey records. In a sense, this experiment is double blind because the generator does not know how the detection method works and the detection method does not know which records are potentially misrepresented. A double blind set up is useful to maintain integrity of the test and prevent developers from engineering records that will be successfully detected by an edit they also created.

Generating this dataset is a medium complexity case because we care about the distribution and magnitude of the dollar amounts. Below is an example of a clean dataset with no errors introduced.

Table I. Four Example Synthetic Records

ID	SECTOR	REVENUE	PAYROLL	EMPLOYMENT
10000	RETAIL	102	41	2
10001	RETAIL	101,005	62,034	780
10002	RETAIL	1,203	320	3
10003	WHOLESALE	304	120	5

To simulate a dataset like this, requirements should be collected about each of the fields such as data type, expected data distributions, and any relationships between the variables. In this case, ID might need to be stored as a string despite looking like a numeric field and should be unique. The sector might come from a collection of 13 economic sector classifications spanning retail and wholesale, but also potentially manufacturing, healthcare, or finance. The economic variables of Revenue, Payroll, and Employment are all numeric but perhaps there are relationships that exist between them. For example, the revenue to payroll ratio might be normally distributed with a mean of 2.5 and a standard deviation of 0.8. Rules like these are important for writing the necessary code to produce these numbers. The technical process of simulating any such variable and the order in which they are simulated is as simple as defining the objective distribution from which the variables are derived then sampling values from those distributions in the appropriate order to preserve stated dependencies.

One strategy to produce revenue and payroll numbers for the dataset in table 1 is to first generate random Revenue values for a set number of records sampled from an even or uniform distribution with a lower bound of 1 and an upper bound of 500,000. This implicitly assumes that your synthetic data will not contain any record with over $500,000,000 in revenue. This upper bound can be adjusted to suit the needs of any specific dataset. Additional restrictions could be imposed such as smaller firms being proportionally more common and therefore, given the number of generated records, a proportion could be set aside for small, medium, and large firms with weights 0.80, 0.15, and 0.05. This latent

identification would direct the generator to sample from different ranges within the specified distribution. Once firm revenues are generated, given the requirement that revenue to payroll ratios must follow a normal distribution with mean 2.5 and standard deviation 0.8, ratios can be produced first from the provided distribution, and payroll values derived from these synthetic constructions.

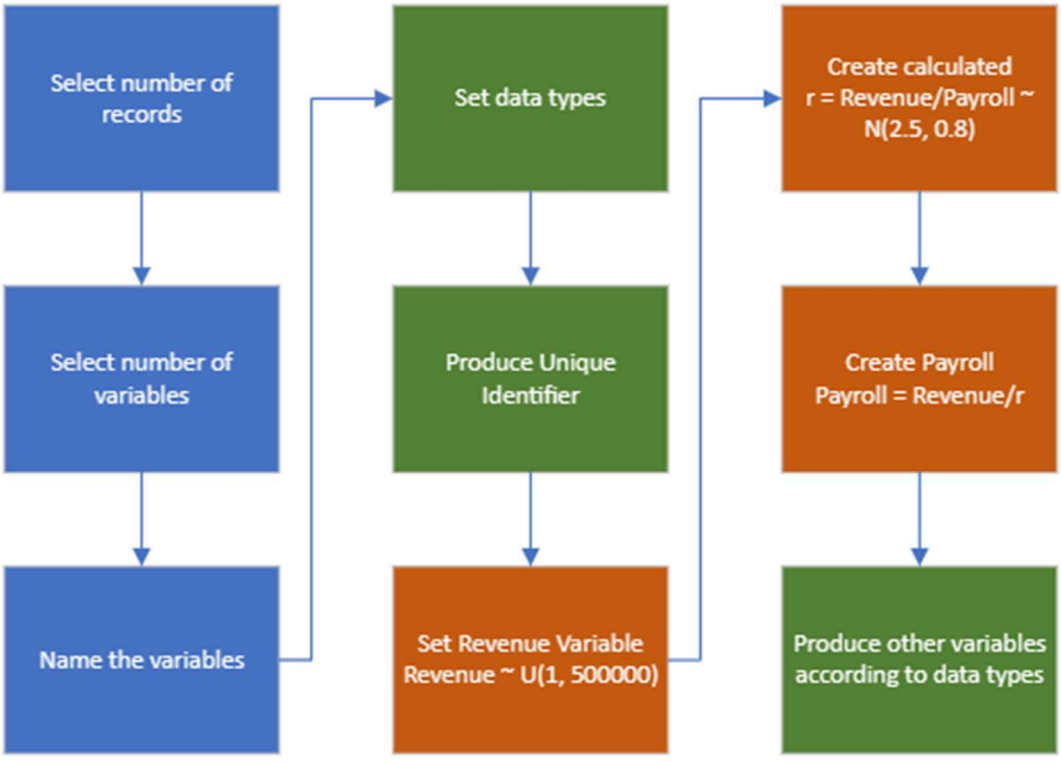
$$Revenue \sim U(1, 500000)$$

$$r = \frac{Revenue}{Payroll} \sim N(2.5, 0.8)$$

$$\frac{Revenue}{r} = Revenue \cdot \frac{Payroll}{Revenue} = Payroll$$

The generation strategy can be mapped out in a tree diagram to better understand the relevant dependencies defined for the system. For this example, the dependencies are simple, but they could become very complex depending on the needs of the user. Figure 1 demonstrates the rules-based dependency tree for this example. The orange and green boxes are order-dependent generation. That is, the production of the identifier and other variables depend on data types being set, and the calculation of r and Payroll first requires the Revenue variable to be constructed.

Figure I. Rules-Based Dependency Tree



Once the clean data are produced through this method, errors can be strategically introduced to allow for the testing of the process. For this exercise, record 10002 will be assumed to have entered the whole dollar amount for their survey response changing the data to the representation in table II.

Table II. Four Example Synthetic Records with Seeded Error

ID	SECTOR	REVENUE	PAYROLL	EMPLOYMENT
10000	RETAIL	102	41	2
10001	RETAIL	101,005	62,034	780
10002	**RETAIL**	**1,203,162**	**320**	**3**
10003	WHOLESALE	304	120	5

One method for finding this relies on the known distribution of the revenue to payroll ratio which comes from the provided generation requirements but could be estimated from historical data. Computing ratios for each record in the table reveal a r value of approximately 3760 for the highlighted record which is presumed to have been drawn from a normal distribution with a mean of 2.5 and a standard deviation of 0.8. If that were true, the resulting Z-score would be 4697, suggesting that this record deviates significantly from the expected distribution. Applying a divide-by-1000 correction to Revenue to adjust the data to the correct units produces a Z-score of 1.575 which is much closer to the known data generating process than the revenue value with erroneous units.

The key challenge with a rules-based generation process is the collection and specification of the rules and other requirements. In the case of data that is yet to be collected, there might not be any historical data to estimate distributions from which makes creating realistic data difficult. There may be other surveys or datasets that capture analogous information from which reasonable parameters might be derived. For example, both the Consumer Expenditure Surveys and the BLS and the American Community Survey at U.S. Census collect income, so using the distribution in any of these surveys for a synthetic household income construct may be a reasonable solution. The other salient consideration is that using data protected under title XIII or XXVI to derive exact distributions for a synthetic data system may create disclosure concerns. It is best to use publicly available microdata or the opinion of subject matter experts on what reasonable distributions look like, when possible, to avoid any issues in this domain. Overall, a method like this one is likely a good fit for when medium or high complexity is desired, especially when complete control over the outcomes is preferred.

3.2 Model-based Generation

Another way of approaching the generation of synthetic data is through model-based methods. Model-based methods revolve around training a statistical model using real data to try and produce data that looks and behaves like the training data. While many such methods can be used for this purpose, this paper discusses two—generative adversarial networks (GANs)[6] and generative pretrained large language models (LLMs).

3.2.1 Generative Adversarial Networks

In traditional artificial neural network model training, the goal is to find a set of parameters through backpropagation[7] such that a loss function (e.g., mean-squared error, cross-entropy (C-E) loss, Wasserstein loss, etc.) is minimized. A GAN differs from this strategy and is 'adversarial' because it sets up two competing networks that try to outperform each other, analogous to an arms race. One network

[6] https://www.researchgate.net/publication/263012109_Generative_Adversarial_Networks
[7] https://neptune.ai/blog/backpropagation-algorithm-in-neural-networks-guide

is the generator, and the other is the discriminator. The training data informs the kind of records that might be produced by the generator. The goal of the discriminator is to determine whether a random record is from the training data or a product of the generator model. This process repeats until eventually the discriminator is unable to determine whether a record is synthetic or real.

To understand how the generator might create new records at scale, a real example from a publicly available federal dataset is useful to consider. The Bureau of Labor Statistics (BLS) produces a public-use microdata (PUMD) file for the Consumer Expenditure (CE) Surveys.[8] The family file is produced quarterly and represents the consumer units (i.e., households) that respond to the survey. Each row of this dataset contains characteristic information about the reference person such as age or race, information about the household itself, like whether it is in an urban or rural area, and information about family income and expenditures. The dataset is remarkably complex and contains rich and varied information about a sample of American households. For showing how GANs work for this process and their potential pitfalls, the family file for 2022 quarter 2 will be considered. The full file contains 5,177 consumer units and 818 variables. A selection of four of those records is displayed in Table III across 5 variables (i.e., a unique identifier, age of the reference person, a flag for age, income, and urbanicity).

Table III. Four Example Consumer Expenditure Interview Survey Records (2022Q2)

NEWID	AGE_REF	AGE_REF_	FINCBTAX	URBAN
4815104	50	D	0	1
4815134	49	D	356411	1
4815284	32	D	138000	1
4815944	87	T	39894	2

For simplicity, the generator and discriminator are trained on all of the records for this subset of variables using a cross-entropy loss function. In these few variables, there are some basic relationships that need to be preserved. For example, the age flag (i.e., AGE_REF_) is equal to 'D' when AGE_REF is a valid, unadjusted value and equal to 'T' when AGE_REF has been 'top coded;' a disclosure avoidance practice where any individual over the age of 87 is set to 87 in the dataset. There is a clear dependency between these variables that would generally require preservation in most synthetic datasets that have any utility beyond load testing. After training the GAN, it was used to produce 1000 new synthetic records. Table IV shows a sample set of the output.

Table IV. Four Synthetic Consumer Expenditure Interview Survey Records from GAN

NEWID	AGE_REF	AGE_REF_	FINCBTAX	URBAN
5815104	64	T	34246	1
5815134	42	D	106445	1
5815284	41	D	-63620	1
5815944	54	D	11485	2

At first glance, the result is reasonable. All the variables have the proper data types and, if categorical, contain the proper categories for that variable based on the underlying training data (e.g., URBAN is

[8] https://www.bls.gov/cex/pumd_data.htm

numeric and contains only 1 and 2 representing Urban and Rural respectively). The most glaring issue is that the model was unable to determine a very simple rule that the age flag should only take a T when age is equal to 87. The underlying training data does not deviate from this rule, but none of the T records in the 1000 synthetic rows have an age value of 87. The first record in table IV is an example of this outcome.

Figure II. Comparison of Age Distribution in GAN Synthetic Records vs. Real CE Data

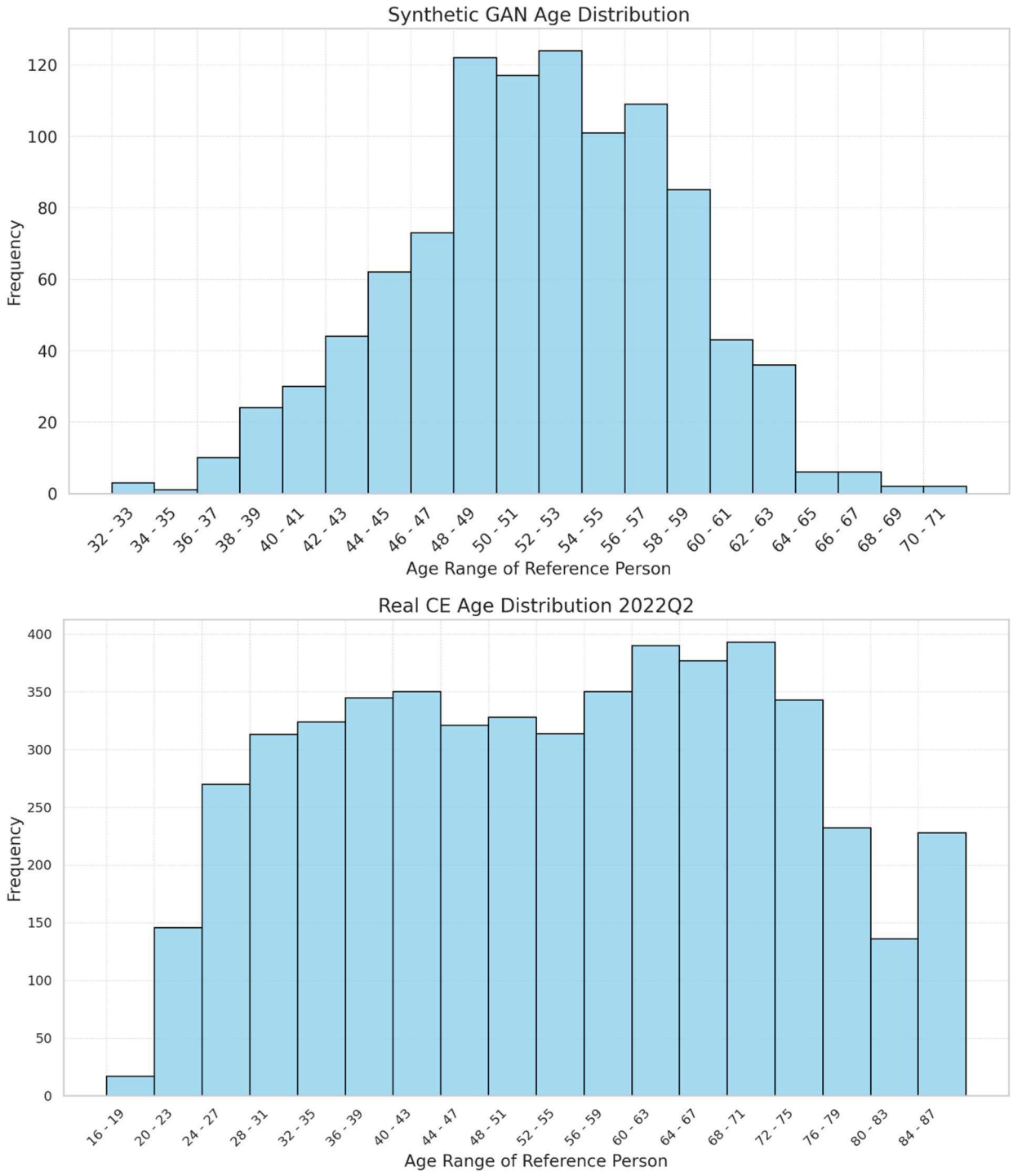
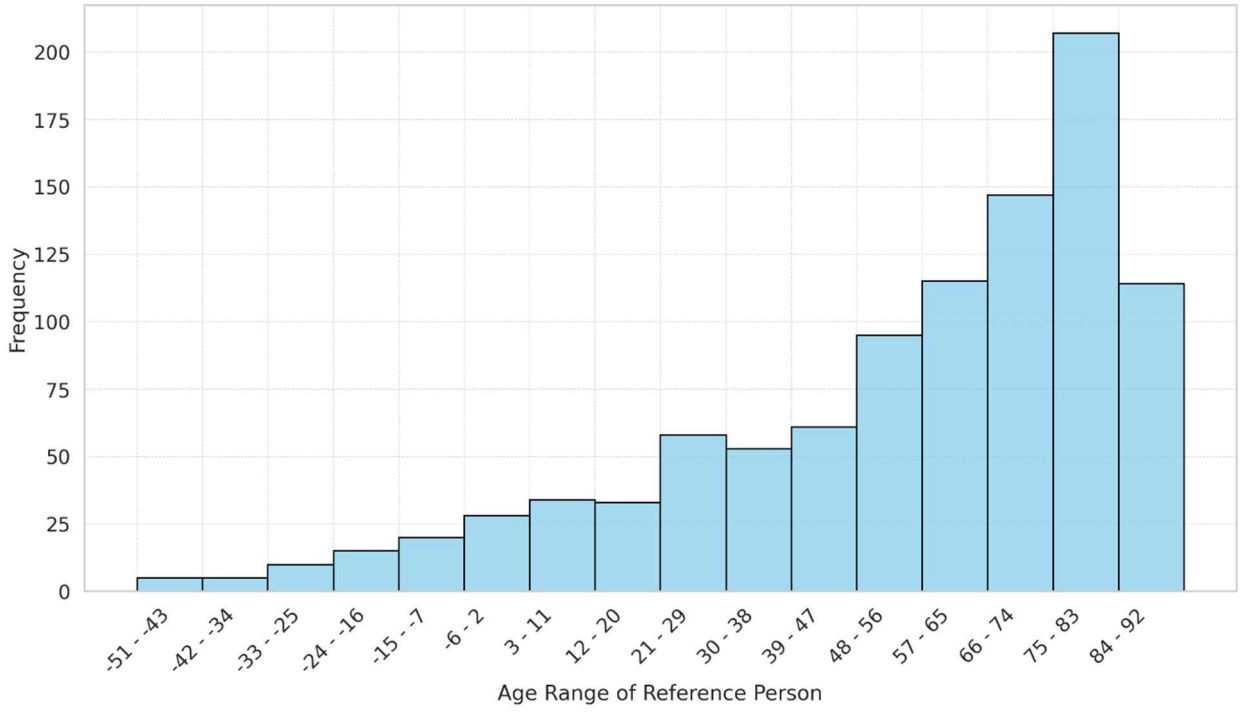


Figure II shows another problem with this generation method called mode collapse, which occurs when the generator model learns a set of outcomes that consistently deceive the discriminator model.[9] As a result, it is disincentivized from deviating from these outputs causing the results of the generation to be too consistent. Put another way, considering one record at a time, it is impossible to determine to which dataset it belongs but when considering the whole set of generated records, the synthetic distribution is a giveaway that these are not the real CE data. The two distributions in Figure II have nearly the same mean, 52 for the synthetic data and 53.8 for the real data. It is a good strategy for the generator to try and pass records close to the mean age in the training data to the discriminator to try and fool it which is why this mode collapse tends to happen for data like these.

There are potential solutions to the mode collapse problem such as different loss functions like Wasserstein loss[10] or using conditional GAN architectures which introduce auxiliary information. Though these alternative loss functions can sometimes produce results that are admittedly more variable, eliminating the mode collapse, they are characteristically worse for realism. When applied to age, the mean of the Wasserstein synthetic distribution is very similar to the mean of the true distribution but has much more variance and contains negative ages. Figure III shows the Wasserstein Loss applied to the age distribution.

Figure III. Age Distribution in Wasserstein Loss GAN Synthetic Data



For categorical variables like Urban, the resulting distributions are reasonably close between the real and synthetic data though the Wasserstein loss performs significantly worse than cross-entropy loss, as shown in table V.

[9] https://link.springer.com/chapter/10.1007/978-3-030-86340-1_45
[10] http://cbcl.mit.edu/wasserstein/wass_NIPS2015.pdf

Table V. Urbanicity by Dataset Crosstabulation

	C-E Synthetic (n=1000)	Wasserstein Synthetic (n = 1000)	Real (n=5177)
1 (Urban)	0.798	0.697	0.810
2 (Rural)	0.202	0.303	0.190

The final example from this experiment shows how certain rare outcomes in the training data may become overrepresented in the synthetic data. It is possible for family income before taxes (i.e., FINCBTAX) to be negative, but it is an extremely rare outcome. Table VI shows how the distributions diverge.

Table VI. Signed Income by Dataset Crosstabulation

	C-E Synthetic (n=1000)	Wasserstein Synthetic (n = 1000)	Real (n=5177)
Negative Income	0.414	0.387	0.001
Positive Income	0.586	0.613	0.999

GANs suffer from potential problems that may conflict with the goals of synthetic data in the federal context. An obvious problem is that, like most statistical models, real data is required as an input. If these data are title protected, it may directly conflict with the agency's policies surrounding their use as an input to a machine learning system. Agencies might also not be comfortable with the highly variable and uncontrolled outputs that GANs can produce.

3.2.2 Large Language Models

Another model-based method involves leveraging the generative pre-trained transformer (GPT) models developed by OpenAI. These large language models (LLM) can take prompts which are fed into their complex weights to produce human-like language and interpretation. The standard GPT models like GPT-4[11] are enormous neural networks trained on much of the written human language on the internet and can be accessed with an application programming interface (API). When provided with a sample dataset[12] and explanation of its content, the GPT models can interpret them, derive variable distributions, and consider them when creating new data, sometimes resulting in impressive outcomes. A major risk for engaging with these kinds of models are 'hallucinations,' a process whereby the LLM produces content that is unconnected, contrived, or incorrectly associated with the input data or request.[13] Like the GAN strategy, 1000 Consumer Expenditure Interview Survey records were generated with GPT-4. An example of some select records are presented in table V.

[11] https://arxiv.org/pdf/2303.08774 - GPT-4 Technical Report

[12] It is best to pass example records to GPT-4 in a structured JSON format when using the API. For the chat interface, a file can be uploaded directly.

[13] https://arxiv.org/abs/2311.05232 - A Survey on Hallucination in Large Language Models

Table V. Four Synthetic Consumer Expenditure Interview Survey Records from GPT-4

NEWID	AGE_REF	AGE_REF_	FINCBTAX	URBAN
4855031	87	D	175524	1
4938353	32	D	-12371	1
4992115	36	D	37253	1
5815944	54	D	148977	2

The model was able to capture all the correct data types for the variables and did not produce any flags that were out of the possible outcomes for urbanicity or the age of the reference person flag. However, even when provided with a strict definition of the flag, the model was unable to apply it correctly. Record 4855031 is a top-coded record and should have a flag of T to reflect this. The LLM could not preserve this relationship and indeed only produced D flags for all records. Tables VII and VIII reflect some of the other tested relationships compared to the real data and the GAN generated synthetic data.

Table VII. Urbanicity by Dataset Crosstabulation

	C-E Synthetic (n=1000)	GPT-4 Synthetic (n = 1000)	Real (n=5177)
1 (Urban)	0.798	0.799	0.810
2 (Rural)	0.202	0.201	0.190

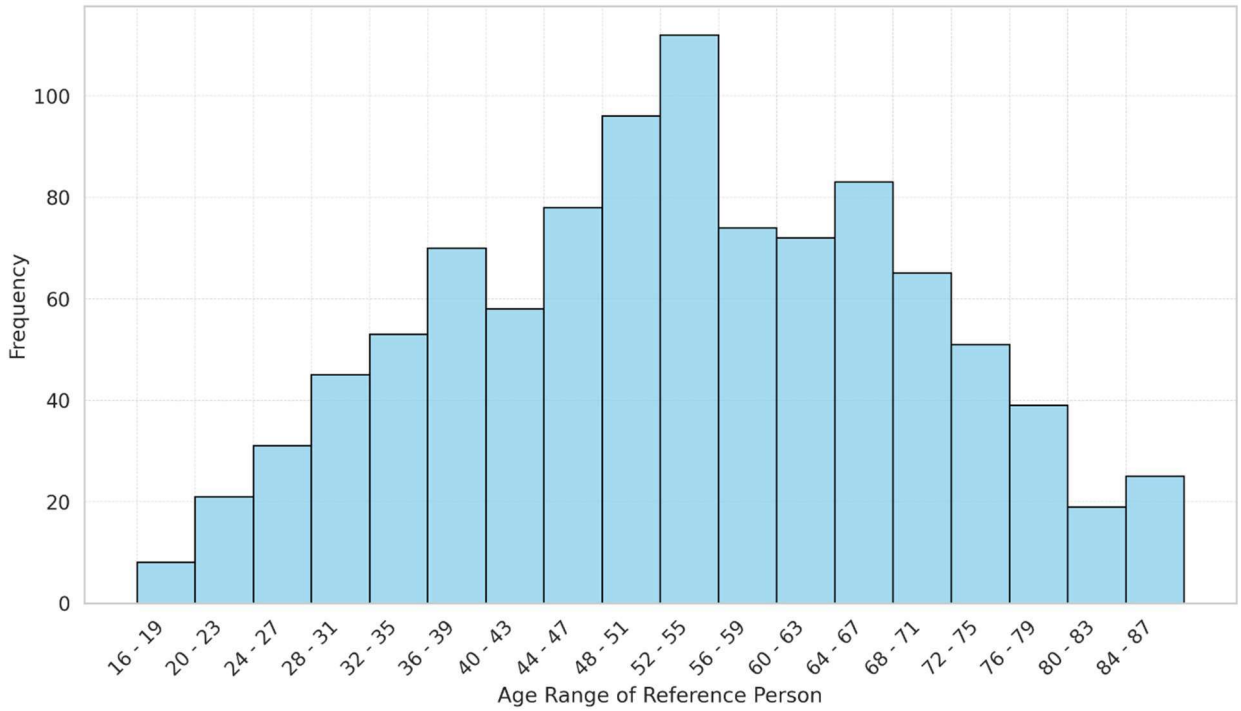
Table VIII. Signed Income by Dataset Crosstabulation

	C-E Synthetic (n=1000)	GPT-4 Synthetic (n = 1000)	Real (n=5177)
Negative Income	0.414	0.137	0.001
Positive Income	0.586	0.863	0.999

These tables show some of the benefits of using GPT-4 to analyze distributions for generation. The urbanicity variable had nearly the same distribution as the GAN generated data and closely reflects the real distribution. In the negative income experiment, the GPT-4 data does a better job than the GAN at approximating the income distribution but still falls short of accurately reflecting the rare nature of a negative income in the real data.

In the case of age, the GPT-4 model does a significantly better job at approximating the age distribution in the base data, preserving the mean and avoiding negative ages. The LLM was not able to determine the over representation of top-coded values present in the real data and the records it generated at the top-coded upper bound failed to have the appropriate flag. Figure IV represents this distribution.

Figure IV. Age Distribution in GPT-4 Synthetic Data



These models do show some improvements over the GAN methodology in terms of distributional accuracy, but they still fall short of being able to identify some of the simple rules present in the input dataset even when specifically prompted by them. Like GANs, they also suffer from requiring real input data to work.

The meticulous level of specificity of a rules-based generator is hard to achieve with model-based methods, especially when trying to replicate distributions or error types without subsequent modification. Because model-based methods employ statistical models, which often include stochastic elements, they introduce a level of unpredictability in the output. This inherent randomness means there is always a risk of unintended errors, which can be both a feature and a bug, depending on the use case, since these models sometimes produce data that is loosely tied or even unconnected to the input parameters. These inaccuracies, while potentially problematic for precise testing scenarios, can be useful for stress-testing systems against unexpected data scenarios. Generally, these model-based results can certainly handle low complexity data and potentially some medium complexity cases where precise control is not necessary.

IV. Mixed Method Generation

It is evident that some of these methods may be well suited for certain tasks in the synthetic generation process. One way to leverage the strengths of the different methods is to combine them. For example, a GPT-4 powered synthetic data set may be considered a first pass generation, then a set of business rules could be applied after the fact to enforce rule requirements that matter. In the case of the Consumer Expenditure Interview Survey dataset, if it is important to the stakeholders that the top-coded flag be correctly applied, the GPT-4 generated data could be checked for ages greater than or equal to 87 and enforce the flag to a ‘T’ with a rule. This merging of rules-based checks with model-based synthesis may

reduce the upfront cost of needing to think about every element of the dataset in advance. Because the model-based methods can generate a lot of variables reasonably well, it is possible to produce large datasets at scale and then selectively apply rules to the variables a researcher or analyst is interested in to ensure correctness, consistency, and applicability to the use case.

Another useful way to employ mixed methods generation is with complex text fields that are common in many federal datasets. For example, a survey may ask an open-ended question about a respondent's survey burden. It may be in the survey methodologists' interest to experiment with different methods of analyzing this kind of text response. In a rules-based generator, often Lorem Ipsum style text[14] is put in place of text responses but these lack the realism necessary to test analysis methods that a researcher might expect to use on an open-ended response to a real survey. By creating data with a rules-based generator then engaging a large language model to generate a column of survey responses to open ended questions, we can increase the realism and applicability of the synthetic data. Table IX shows an example of LLM-assisted synthetic data.

Table IX. Five Synthetic Consumer Expenditure Interview Survey Records from GPT-4 with Burden

NEWID	AGE_REF	AGE_REF_	FINCBTAX	BURDEN
4865031	34	D	185120	Completing this survey was quite straightforward and didn't take much time at all.
4930353	22	D	-71	I found the survey to be a bit lengthy and confusing at times but was able to answer everything.
4991115	43	D	57166	It was moderately challenging due to the detailed answers required.
5015944	74	D	28002	I felt overwhelmed by the number of questions and the complexity of some topics.
5021122	87	T	10112	The survey was not burdensome; the questions were clear and easy to answer

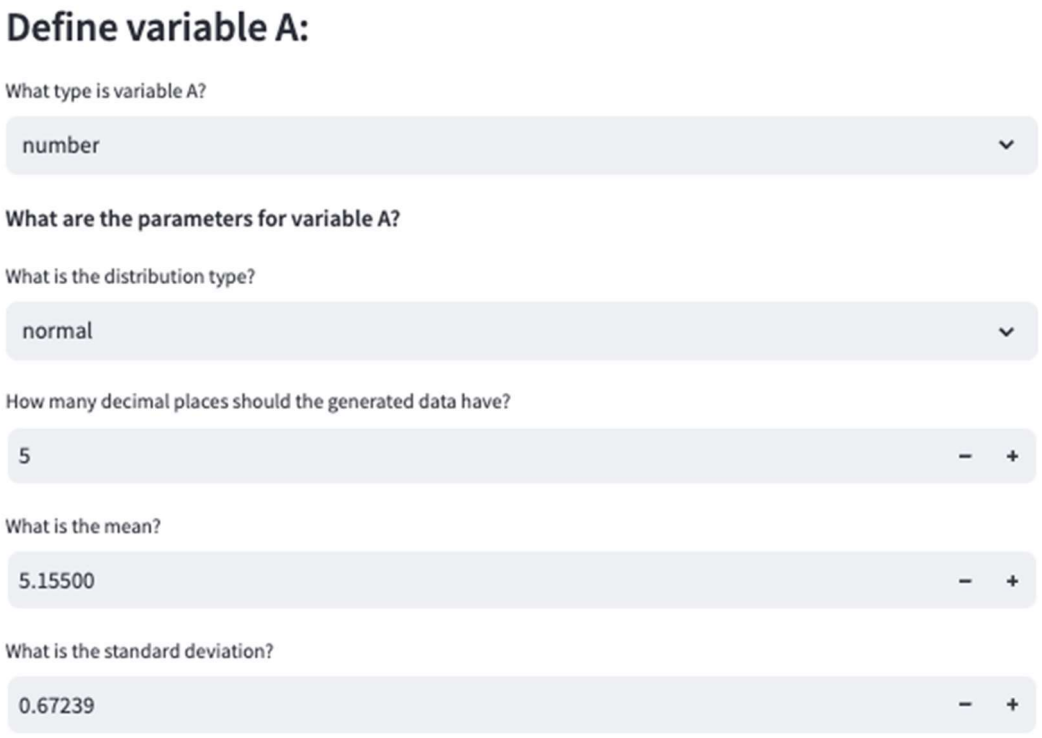
[14] https://www.lipsum.com/ - Information on Lorem Ipsum

V. Requirements Gathering Platform

Despite the control rules-based generation provides, it still has inherent challenges. After a synthetic data user identifies the required level of complexity necessary to effectively accomplish their intended task, they must begin the tedious process of collecting and precisely specifying the rules and requirements to define the rules-based generation process. Additionally, amidst the many possible decisions surrounding specification, discerning what is pertinent information to include in the generation process is essential to producing high-quality synthetic datasets. These upfront costs may be prohibitive for very complex synthetic data needs. This highlights why one may choose to use a model-based generator which has very low upfront costs to generate many records at scale.

One solution to these challenges is a requirements gathering platform, a tool designed to streamline the rules-based data generation process and enhance communication between synthetic data developers and project owners. The platform allows for the creation of inputs to rules-based generation software. Intuitive prompts and tailored inquiries guide users to formulate inputs that align with their goals. The platform allows users to specify the number of variables to define and the number of datapoints to generate, name variables, specify the variable types, and describe the parameters of each variable. After reviewing their selections, users can create and download their synthetic data. Figure V provides an example of this guidance.

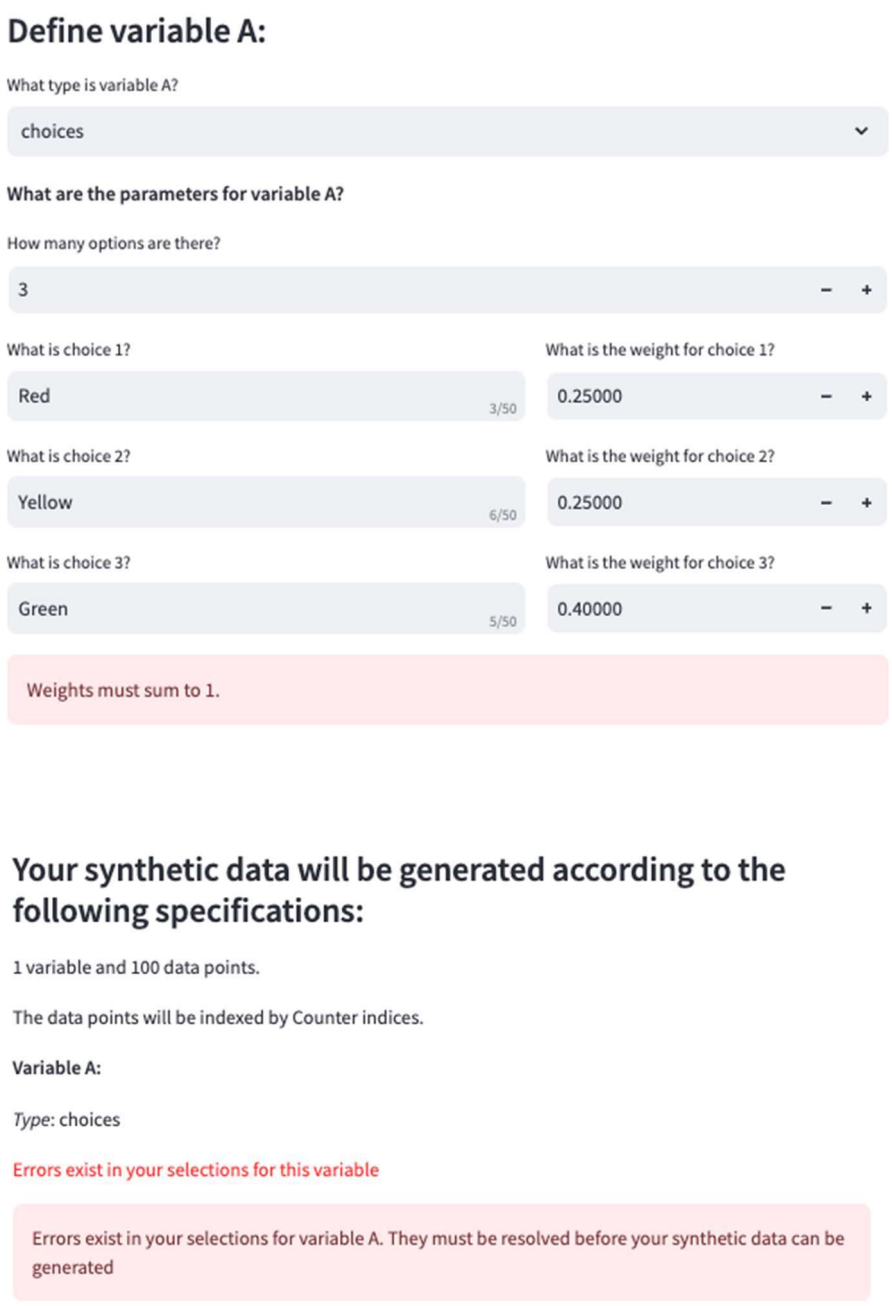
Figure V. Prompts in a Requirements Gathering Platform



The platform seamlessly combines user input with the generation software to create synthetic data quickly and easily, without requiring extensive technical expertise. Furthermore, the platform safeguards against logical inconsistencies by checking inputs. By ensuring the coherence and correctness

of user-provided specifications, it guards against erroneous or undesired outputs. Figure VI shows the guidance the platform can provide to protect against errors.

Figure VI. Errors Presented During the Specification Process



The creation of a requirements gathering platform helps overcome some of the challenges of a rules-based generation process by bridging the gap between synthetic data developers and users. It minimizes the amount of time needed to gather specifications by guiding the user to provide informative requirements surrounding their synthetic data needs. Additionally, if the requirements gathering platform proves too limiting for the complexity of the desired synthetic dataset, users are alerted to the need for a higher level of complexity. In these cases, users can work with developers directly to accomplish their goals. Through an intuitive interface, the platform empowers users to navigate the intricacies of data specification and preempts potential errors, offering corrective feedback to ensure precision and accuracy. This platform reduces the setup cost and effort required to use rules-based generation by streamlining communication and guiding non-technical users in how to properly communicate rules and requirements.

VI. Conclusion

The creation of synthetic data revolves around the selection between rules-based and model-based generation methods, contingent upon the desired complexity and accuracy of the resulting dataset. For use cases that require a medium to high level of complexity with meticulous control over data outcomes, rules-based generation stands out as the method of choice. It provides a structured and deterministic approach to data creation, ensuring that every piece of synthetic data adheres to predefined rules and distributions. This method is particularly advantageous when precise, controllable inputs are necessary for testing environments, and when the use of sensitive, real-world data is restricted or poses potential disclosure risks. However, the challenge lies in defining these rules without relying on protected data, which necessitates innovative solutions such as utilizing public microdata or leveraging the expertise of subject matter experts to approximate realistic distributions. The requirements gathering platform assists those experts with providing all the necessary parameters to the rules-based generator so that the resulting data set can be constructed accordingly, lowering the transaction costs of the synthesis.

In contrast, model-based methods are well-suited to scenarios that benefit from an element of unpredictability, reflecting the inherent variability and randomness found in actual data environments. While these methods may not offer the same level of precision as rules-based generation, they excel in creating diverse datasets that can expose systems to a wide range of data scenarios, including those that are unforeseen but plausible. This capability is particularly useful in stress-testing and enhancing the resilience of systems. Though model-based synthetic data may be more aligned with lower complexity requirements, with careful calibration, it can also address some medium complexity needs where exact control over data is less critical. Ultimately, the decision on the generation method depends on a careful evaluation of the specific requirements of the system being tested or developed, with the goal of striking a balance between control, realism, and the protective mandates governing the use of sensitive data.

By acknowledging the strengths and limitations of each synthetic data generation approach within the established complexity framework, organizations can make informed decisions that align with their operational needs and compliance obligations. The thoughtful application of these methods will pave the way for more secure, efficient, and reliable data processes within the constraints of privacy and regulation.