Contributions to Statistical Theory of Data Privacy

by

Chang Qu

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.Sc. degree in
Mathematics and Statistics[1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1]The MSc program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics.

Contents

Chapter 1

Introduction

1.1 Data Privacy

We use data in every aspect of our lives. Having data helps researchers to test new drugs or companies to better target advertising campaigns. However, data may also contain private information which causes serious privacy problems. Even if data is somehow anonymized, as data becomes more detailed, there's a higher risk of revealing personal information, which can result in misuse, identity theft, discrimination, and a loss of public trust. A big challenge is to balance **data privacy** with **data utility**.

This thesis, funded by MITACS and an Ottawa-based company Privacy Analytics, deals with several practical questions that the company needs to address when dealing with clients. We provide explanations and insights into several disclosure risk measures, such as a widely-used Correct Attribution Probability (CAP). We show that some of these measures lack interpretability or have a limited applicability. As such we provided some solutions. Additionally, we reviewed the methods for synthetic data generation. Many of these methods are used by practitioners to create ”private” data. Again, we indicate several issues with different approaches and propose solutions.

Disclosure risk measures. To be more specific, despite the importance of disclosure risk measures like Correct Attribution Probability (CAP), explaining them to clients is a significant challenge. Clients need to understand the risks of data disclosure in simple and clear terms to make informed decisions about how to handle data and protect privacy. This thesis focuses on developing effective ways to explain disclosure risk measures, especially CAP, to clients in an easy-to-understand manner. It also addresses the limitations of the traditional CAP by proposing a new modified CAP. This new approach is based on equivalence classes, has a proper interpretation and can be applied to both discrete and continuous data.

Synthetic data generation. Additionally, the thesis explores ways to ensure that synthetic data are similar to the original data in terms of statistical properties and usefulness. Synthetic data generation is a key technique in data privacy, it is a form of data anonymization by creating entirely new datasets that replicate the statistical characteristics of the original data. This approach ensures that no original data is released, thereby minimizing the disclosure risk while still allowing meaningful data analysis. The methodologies discussed here include the Inverse Transformation Method, Multivariate Inverse Transform Sampling, Acceptance-Rejection Method, Bootstrapping, and the Synthetic Minority Over-sampling Technique (SMOTE) and interpolation methods. As the names suggest, some of these methods are just traditional simulation methods, while others are less known. One has to mention that traditionally the synthetic data generation techniques focus on data utility, with less focus on data privacy. We brought the latter aspect to our analysis.

Synthetic data generation - *Synthpop* package. In the context of synthetic data generation, one of the key focus of this thesis is the detailed examination of a software package called *Synthpop*. This package is widely used in the community. The detailed analysis covers its algorithms, including various models used to generate synthetic data. However, the thesis also highlights current limitations of *Synthpop*, such as challenges in maintaining dependency structures between variables and handling complex dataset. For instance, the package may struggle to preserve complex inter-variable relationships, leading to synthetic data that does not accurately reflect the original dataset's structure and statistical properties. Moreover, the thesis critiques the utility metrics used by *Synthpop*, pointing out that some of the statistical tests, like the Kolmogorov-Smirnov test, may be incorrectly applied. These tests assume independence between samples, which is not the case when comparing original and synthetic datasets, potentially resulting in misleading conclusions about data similarity.

To address these issues, the thesis proposes several improvements. Instead of generating variables sequentially, each variable is modeled using all other variables as predictors, with appropriate model selection tools. This approach helps preserving dependencies without relying on a fixed generation order. Furthermore, the thesis introduces more appropriate statistical tests and metrics that account for the dependencies between datasets, ensuring a more accurate assessment of data utility.

Furthermore, the package focuses on data utility of generated datasets, ignoring their privacy. Therefore, in the thesis we address the data privacy issues using the modern tools such as differential privacy. With its help, the synthetic data generation process provides stronger privacy guarantees while maintaining data utility.

By identifying and addressing these limitations, the thesis aims to improve the effectiveness of *Synthpop* and similar tools in the generation of high-quality synthetic data. This ensures that synthetic datasets are not only useful for analysis but also robust in protecting individual privacy. These enhancements contribute to the overall goal of balancing data utility with privacy, providing a reliable tool for data sharing and analysis without compromising the confidentiality of sensitive information.

1.2 Thesis Structure and Descriptions

In Chapter 2 we explore disclosure risk measures. We start with the basic terminology, such as the notion of a sensitive attribute or a quasi-identifier.

- Section 2.1 discusses basic metrics for assessing disclosure risks such as the proportion of unique entries and their relative frequencies within datasets. These metrics apply to discrete data.
- Section 2.2 explores k-anonymity as a privacy-enhancing technique or as a privacy measure that protects against data re-identification by ensuring that each record is indistinguishable from at least $k - 1$ other records. To achieve this, the data are grouped in equivalence classes.
- Section 2.3 discusses ℓ-diversity, a method that provides stronger than k-anonymity guarantees by ensuring that each equivalence class has at least ℓ well-represented, distinct sensitive values, thus preventing attribute disclosure.
- Section 2.4 examines t-closeness which requires the distribution of a sensitive attribute within any equivalence class to closely resemble the overall distribution for that attribute across the entire dataset.
- Section 2.5 explores differential privacy, a more advanced mathematical framework to quantify the privacy techniques used to ensure that the results of the analyzes are not overly dependent on any individual record in the data set.
- Section 2.6 provides detailed analysis of Correct Attribution Probability (CAP), a measure that assesses the risk of an attacker correctly re-identifying sensitive information through statistical inferences. To be more specific:
  - Section 2.6.2 discusses the original CAP scores, their computation, implications for data privacy, and their limitations, particularly when dealing with continuous data.
  - Section 2.6.3 and Section 2.6.4 introduce a new approach to CAP by using equivalence classes, making it easier to assess complex real-world datasets.
  - Section 2.6.5 explores CAP from a population-level perspective, offering a measure on how well entire datasets protect against inference attacks.
  - Section 2.6.5 compares the disclosure risks associated with the original and anonymized datasets, highlighting the effectiveness of applied anonymization techniques.

  - Section 2.6.6 explores how adding noise, intended to increase privacy, impacts CAP scores. We also reveal that the scores vary depending on the chosen equivalence classes.

In Chapter 3 we explore some methods for synthetic data generation, starting with the basic ones (well-known in the probabilistic and statistical communities), and then presenting less known ones.

- Section 3.2 explains how to create synthetic data when we know the exact data distribution.
- Section 3.2.1, Section 3.2.2, Section 3.2.3 discuss inverse transform sampling for creating data that follows a specified distribution, suitable for both continuous and discrete data.
- Section 3.2.4 introduces the acceptance-rejection method for sampling from complex distributions.
- Section 3.2.5 and Section 3.2.6 describe how data transformations, conditioning and mixture distributions are applied to achieve specific properties in synthetic data.
- Section 3.2.7 focuses on using parametric methods.
- Section 3.2.8 explains how to create multivariate distributions using a decomposition.
- Section 3.2.9 explores the bootstrap method for resampling data to create new datasets.
- Section 3.2.10 compares inverse transformation and bootstrap techniques.
- Section 3.2.11 introduces the Synthetic Minority Over-sampling Technique (SMOTE), a machine learning method for generating synthetic samples which can address class imbalance.
- Section 3.2.12 discusses linear interpolation techniques for generating data points from observed data points.

In Chapter 4, we detail the *Synthpop* software package, focusing on its description, challenges, and solutions:

- Section 4.2 explains the simulation algorithm used by the package, detailing various model selection options.
- Section 4.3 discusses how the *Synthpop* package handles missing data and correlation between variables.

- Section 4.4 examines how the package measures the data utility of synthetic data while pointing out limitation of the measures provided.
  - Section 4.4.1 focuses on the use of the Mean Squared Error and coefficient of determination to evaluate the accuracy and reliability of synthetic data.
  - Section 4.4.2 and Section 4.4.3 cover the comparison of marginal empirical cumulative distribution functions and the use of contour plots to assess the similarity between original and synthetic data distributions.
  - Section 4.4.4 explains how propensity scores are used to assess the probability of each data point being synthetic or real, aiding in utility and privacy evaluations.
  - Section 4.4.5 looks at how contingency tables are utilized to compare categorical data across original and synthetic datasets.
  - Section 4.4.6 discusses permutation tests used to statistically compare the distributions of original and synthetic data.
  - Section 4.4.7 explores how the Gini index is applied to measure inequality in the distribution of attributes between original and synthetic datasets.
- Section 4.5 addresses specific challenges and limitations encountered when using the *Synthpop* package, providing solutions and workarounds.
  - Section 4.5.1 details how the *Synthpop* package relies on the sequential relationships in the original data, we also demonstrates that incorrect sequential relationships can lead to errors.
  - Section 4.5.2 describes scenarios where no sequential relationships exist, leading to potentially lower utility in generated data. We also propose a method to address this issue.
- Section 4.6 proposed a differential privacy method for data generation that relies on statistical models.

1.3 Thesis Contributions

This thesis contributes to the field of data privacy and synthetic data generation by providing statistical and probabilistic interpretations and improvements to existing methodologies. We indicate several issues and propose some solutions. The primary contributions are outlined as follows:

- In Chapter 2, we provide a comprehensive review of current disclosure risk measures such as k-anonymity, l-diversity, t-closeness, and differential privacy, analyzing their applicability and limitations. we also introduced a novel approach for Disclosure

Risk Assessment based on Correct Attribution Probability (CAP), which considers equivalence classes to improve interpretability.

- Chapter 3 reviews various synthetic data generation techniques.
- In Chapter 4 we analyse the *Synthpop* software package, where we introduced new features that address limitations such as selecting the correct sequence and handling non-sequential data relationships. Additionally, we implemented and validated new metrics for evaluating synthetic data. Moreover, we developed a differentially private method of data generation based on statistical models, significantly enhancing privacy while maintaining data utility.

Chapter 2

Disclosure Risk Measures and Anonymization

This chapter explores methods for measuring disclosure risk in databases. We discuss:

- measures based on analyzing data uniqueness across population, sample, and anonymized datasets; see Section 2.1;
- k-anonymity; see Section 2.2;
- ℓ-diversity; see Section 2.3;
- t-closeness; see Section 2.4;
- differential privacy; see Section 2.5;
- Correct attribution probability (CAP); Section 2.6.

These concepts are essential for protecting privacy. We present examples and discuss both applicability and limitations of these measures. Each measure requires a different set-up. It should be pointed out that although measures of uniqueness, k-anonymity, t-closeness, ℓ-diversity and CAP are of similar type, the concept of differential privacy is completely different.

This chapter is based on the existing literature. The reviewed literature includes [11], [36], [34], [31], [3], [13], [16], [29], [9], [10], [19], [17], [5], [3], [28], [14], [11], [11], [33], [20], [8]. Some of the definitions are quite obvious and it is hard to identify when they appeared for the first time in the literature.

2.1 Basic disclosure risk measures

In this section we discuss basic, non-probabilistic, disclosure risk measures, that are randomly scattered through the literature. The list includes:

- proportion of unique entries; see Section 2.1.1.
- Relative frequency of unique entries; see Section 2.1.2.

2.1.1 Proportion of unique entries

These measures typically consider uniqueness of particular records. As such, they are typically applicable to discrete data. The main idea behind the following definition is that data with unique quasi-identifiers are considered risky, as the attacker can identify such individuals with probability 1.

Problem setup: We are given databases: the population database Z, the sample database X and the anonymized database Y.

Definition 2.1.1 (Proportion of Unique Entries)**.** *Consider a dataset* $Q = (Q_1, \ldots, Q_N)$ *of size* N*, indexed by* $U = \{1, \ldots, N\}$*. Assume that we have* J *equivalence classes, denoted as* $[1], \ldots, [J]$*. We define*

$$\mathrm{PUE}(Q) := \frac{1}{J}\sum_{j=1}^{J} 1\left\{f_j(Q) = 1\right\} \ , \tag{2.1}$$

where 1 *is the indicator function and* $f_j(Q) = \sum_{i \in U} 1\left\{Q_i \in [j]\right\}, j = 1, \cdots, J$*.*

It calculates the proportion of units in the given dataset which are unique. The bigger the number is (on the scale [0,1]), the more unique items, the less privacy. The abbreviation PUE(Q) stands for "Proportion of Unique Entries" (of the database Q).

We will apply this definition to either Z (population database), X (sample database) or Y (anonymized database). Note that the equivalence classes and the sample size may be different for each of the databases X, Y or Z. Comparing the PUE numbers between the databases allows us to conclude which one is more private.

Example 2.1.2. For this example,

- The dataset Z consists of 31 people and contains their names, gender and date of birth (DOB). We have many equivalence classes.
- The original database X contains $n = 14$ people with their names, gender, DOB, EIs, QIs, as well as medical test results (SAs).
- The dataset Y has been obtained from X via anonymization procedure (called 2-anonymization, to be discussed later). The names are removed and the gender is kept, while the DOB is replaced with a range of dates. The test results are kept. We have $J = 5$ equivalence classes that correspond to "{Male, [1950-1959]}", "{Male, [1960-1969]}", etc.

Table 2.1: Z

	EI	QI	
ID	Name	Gender	Year of Birth
5	Alicia Freds	Female	1942
31	Natasha Markhov	Female	1941
28	Alex Long	Female	1952
3	Alice Brown	Female	1955
29	Britney Goldman	Female	1956
18	Jane Doe	Female	1961
7	Marie Kirkpatriek	Female	1966
21	Kathy Last	Female	1966
19	Nina Brown	Female	1968
11	Beverly McCulsky	Female	1964
24	Alexandra Knight	Female	1974
6	Gill Stringer	Female	1975
13	Freda Shields	Female	1975
17	Lillian Barley	Female	1978
8	Leslie Hall	Female	1987
30	Lisa Marie	Female	1988
27	Almond Zipf	Male	1954
4	Hercules Green	Male	1959
1	John Smith	Male	1959
12	Douglas Henry	Male	1959
15	Joe Doe	Male	1961
2	Alan Smith	Male	1962
14	Fred Thompson	Male	1967
22	Deitmar Plank	Male	1967
26	Anderson Heft	Male	1968
10	Albert Blackwell	Male	1978
20	William Cooper	Male	1973
23	Anderson Hoyt	Male	1971
9	Bill Nash	Male	1975
16	Mark Fractus	Male	1974

Table 2.2: X

	EI	QI		SA
ID	Name	Gender	Year of Birth	Test Result
5	Alicia Freds	Female	1942	- ve
3	Alice Brown	Female	1955	- ve
11	Beverly McCulsky	Female	1964	- ve
7	Marie Kirkpatrick	Female	1966	Zero
13	Freda Shields	Female	1975	- ve
6	Gill Stringer	Female	1975	- ve
8	Leslie Hall	Female	1987	- ve
4	Hercules Green	Male	1959	- ve
12	Douglas Henry	Male	1959	+ ve
1	John Smith	Male	1959	+ ve
2	Alan Smith	Male	1962	- ve
14	Fred Thompson	Male	1967	- ve
9	Bill Nash	Male	1975	- ve
10	Albert Blackwell	Male	1978	- ve

Table 2.3: Y

	QI		SA
ID	Gender	Decade of Birth	Test Result
13	Female	1970-1979	-ve
6	Female	1970-1979	-ve
11	Female	1960-1969	-ve
7	Female	1960-1969	Zero
12	Male	1950-1959	+ve
1	Male	1950-1959	+ve
4	Male	1950-1959	-ve
2	Male	1960-1969	-ve
14	Male	1960-1969	-ve
9	Male	1970-1979	-ve
10	Male	1970-1979	-ve

For the database Z, we consider the Equivalence Class = {Gender, Year of Birth}. We have:

- for $j = 1$, {Female, 1942}, the equivalence class is unique in the population data, thus we have $f_1(Z) = 1$;
- for $j = 2$, {Female, 1941}, the equivalence class is unique in the population data, thus we have $f_2(Z) = 1$;
- similarly, for $j = 3$ to 25, except for $7, 11, 16$ and 19, we also have $f_j(Z) = 1$;
- 21 out of 25 equivalence class are sample unique, thus
$$\text{PUE}(Z) = \frac{21}{25} .$$
This number is high, close to 1. This is concerning from the point of view of privacy and some anonymization is needed.

Further, we consider the sampled dataset X. Here, we have the same Equivalence Classes as for Z.

- for $j = 1$, the equivalence class {Female, 1942} is unique in the sample data, thus $f_1(X) = 1$;
- similarly, we have $f_j(X) = 1$ for $j = 1, 2, 3, 4, 6, 8, 9, 10, 11$;
- for $j = 5$ {Female 1975}, there are two samples for this equivalence class, $f_5(X) = 2$;
- for $j = 7$ {Male 1959}, there are three samples for this equivalence class, $f_7(X) = 3$;
- hence,
$$\text{PUE}(X) = \frac{11}{13} .$$
The sampled data has almost the same level of privacy as the external data Z.

Now, we consider the anonymized database Y. Here, Equivalence Class = {Gender, Range for Year of Birth}.

- for $j = 1$ and {Female, [1970-1979]}, there are 2 data in the sample that have the same equivalence class, thus we have $f_1(Y) = 2$;
- for $j = 2$ and {Female, [1960-1969]}, there are 2 data in the sample that have the same equivalence class, thus we have $f_2(Y) = 2$;
- Similarly, $f_3(Y) = 3$, $f_4(Y) = 2$ and $f_5(Y) = 2$.

None of the data in the sample is unique in the sample. Hence, the expression in (2.1) is zero. It is an indication that some level of privacy was achieved. □

2.1.2 Relative frequency of unique entries

Once we calculate the number of unique entries, we can relate unique entries between the population and the sampled databases. We will consider the following two scenarios.

Problem setup:

- Scenario 1:
  - The identification dataset Z is given.
  - The sample dataset X is given.
- Scenario 2:
  - The sample dataset X is given.
  - The population dataset is not given. Thus, we need to estimate the number of sample that is unique in the population.

We note that this method requires that the equivalence classes are the same for the databases being compared. For example, dataset X and dataset Y cannot be compared because they have different equivalence classes.

Definition 2.1.3 (Relative Frequency of Unique Entries)**.** *Consider datasets Q_1, Q_2 of size N_1, N_2, indexed by $U_j = \{1, \ldots, N_j\}$, $j = 1, 2$, respectively, with the same equivalence classes, denoted as $[1], \ldots, [J]$. We define*

$$\mathrm{RFUE}(Q_2 \mid Q_1) := \frac{\sum_{j=1}^{J} I\left\{f_j(Q_1) = 1, f_j(Q_2) = 1\right\}}{\sum_{j=1}^{J} I\left\{f_j(Q_1) = 1\right\}} .$$

The above expression is easy to compute when $Q_1 = X$ and $Q_2 = Z$ are given. That is, we compare the sampled database with the population. If the population database is not given, the aforementioned probability can be approximated using the Bayes Rule:

$$\widehat{\mathrm{RFUE}}(Z \mid X) = \frac{\widehat{p}_1 \cdot \mathbb{P}\left(f_j(X) = 1 \mid f_j(Z) = 1\right)}{\sum_{\text{all i}} \widehat{p}_i \cdot \mathbb{P}\left(f_j(X) = i \mid f_j(Z) = i\right)} ,$$

where $\widehat{p}_i$ is the estimated proportion of equivalence classes of size i in the population and $\mathbb{P}\left(f_j(X) = i \mid f_j(Z) = i\right)$, the probability that an equivalence class $[j]$ in the sample has size i, given that it has the same size in the population, can be assumed to follow hypergeometric distribution for all i's. That is,

$$\mathbb{P}\left(f_j(X) = i \mid f_j(Z) = i\right) = \frac{\binom{i}{i}\binom{N-i}{n-i}}{\binom{N}{n}} , \quad j = 1, \ldots, J ,$$

where N is the size of the original population and n the size of the sample.

Example 2.1.4. We continue with data from Example 2.1.2.

- We assume that Z is not given, We first calculate $\widehat{p_j}$ and we estimate $\widehat{\text{RFUE}}$ based on the sampled database X.
  - among 11 equivalence classes ($\{$Gender, DOB$\}$), we have 9 sample unique equivalence classes. Hence, $\widehat{p}_1 = \frac{9}{11}$;
  - we have 1 equivalence class of size 2: $\widehat{p}_2 = \frac{1}{11}$;
  - and we have 1 equivalence class of size 3: $\widehat{p}_3 = \frac{1}{11}$.
- We now use hyper geometric distribution to estimate $\mathbb{P}\left(f_j(X) = i \mid f_j(Y) = i\right)$ for each i:
  - for $i = 1$ we have $\mathbb{P}\left(f_j(X) = 1 \mid f_j(Y) = 1\right) = \frac{\binom{1}{1}\binom{31-1}{14-1}}{\binom{31}{14}} = 0.4516$;
  - for $i = 2$ we have $\mathbb{P}\left(f_j(X) = 2 \mid f_j(Y) = 2\right) = 0.196$;
  - and $i = 3$ we have $\mathbb{P}\left(f_j(X) = 3 \mid f_j(Y) = 3\right) = 0.081$.
- Therefore, our estimated quantity is

$$\frac{0.818 \times 0.4516}{0.818 \times 0.4516 + 0.091 \times 0.196 + 0.091 \times 0.081} = 0.9361 \ .$$

What does it mean? If we see an a unique record in our sample, we are 93% sure that this record is also unique in the population. Thus, the sampled database has little privacy and has to be anonymized. □

2.2 Measuring disclosure risk via k-anonymity

In Section 2.1 we introduced some measures that tell us how private is our database. These measures were primarily applied to the population and sampled databases, Z and X. If there is not enough privacy, we need to perform some anonymization. The simplest approach is grouping and generalization with the associated measure of disclosure risk called k anonymity. The term k-anonymity was first introduced by Pierangela Samarati and Latanya Sweeney in the paper published in 1998 ([27]), although the concept dates to a 1986 paper by Tore Dalenius; [6].

Definition 2.2.1 (k-anonymity)**.** *A database satisfies k-**anonymity** if each equivalence class of Quasi-Identifiers consist of at least k units.*

In principle, k-anonymity should guarantee that the chance of re-identification is at most $1/k$. However, if we possess less information (e.g., we do not know whether the individual is in the dataset), the chance of re-identification is lower. Conversely, if we have additional information (e.g., knowledge of certain quasi-identifiers or demographic details about the individual), the chance of re-identification can increase.

Problem setup: We assume the attacker is applying the homogeneity attack. The attacker has no information about data distribution and only has access to two datasets:

- The dataset X of size n. It contains Explicit Identifiers (EI), Quasi-Identifiers (QI) and Sensitive Attributes (SA). The entries in the database are denoted by $X_1, \dots, X_n$.
- The size of the anonymized dataset Y, m, may not be equal to n. It is obtained from X via k-anonymization. Generalization and suppression are typically employed to achieve k-anonymity. Generalization reduces the detail of quasi-identifiers by grouping specific values into broader categories, making records less distinct and harder to trace to individuals. Suppression involves removing data by deleting specific entries or entire attributes that are too identifying or insufficiently generalized. The entries of the database are denoted by $Y_1, \dots, Y_m$. This dataset is available to the attacker.
- We will denote by $[j]$, $j = 1, \dots, J$, the equivalence classes determined by the anonymized dataset.
- To apply k-anonymization algorithm the equivalence class is determined by original data. To calculate disclosure risk, equivalence class is determined by anonymized dataset.
- If we want to calculate the probability of disclosure we need to consider an external or identification dataset Z. In this context, we are considering the original dataset X with sensitive attributes removed, meaning we know exactly which individuals are in the dataset. If a different external dataset Z is used, the probability of re-identification will differ.

Example 2.2.2. We continue with Example 2.1.2. Recall that

- The database X contains $n = 14$ people with their names (EIs), gender and date of birth, DOB, (QIs) as well as medical test results (SA).
- The dataset Y has been obtained from X via 2-anonymization. The names are removed, the gender is kept, while the DOB is replaced with a range of dates to achieve 2-anonymity. The test results are kept.
- We have $J = 5$ equivalence classes that correspond to ”Male, [1950-1959]”, ”Male, [1960-1969]” etc.

We consider the 2-anonymized database Y. We note that in this example above there is 2-anonymity achieved with respect to Gender and DOB, but not with respect to Test Result. Indeed, if we know that Hercules Green is in the anonymized database (we know his range of DOB), then the chance of guessing his Test Result is $1/2$. However, if we know

Table 2.4: Table Y

	QI		SA
ID	Gender	Decade of Birth	Test Result
13	Female	1970-1979	-ve
6	Female	1970-1979	-ve
11	Female	1960-1969	-ve
7	Female	1960-1969	Zero
12	Male	1950-1959	+ve
1	Male	1950-1959	+ve
4	Male	1950-1959	-ve
2	Male	1960-1969	-ve
14	Male	1960-1969	-ve
9	Male	1970-1979	-ve
10	Male	1970-1979	-ve

that Freda Shields is in the database (again, we know her DOB), then we automatically know what is her Test Result. We have attribute disclosure for Freda Shields! In other words, this example shows that **k-anonymity measures identity disclosure, not attribution disclosure**. Even though we know the sensitive attribute for the individual, the probability of knowing that a particular row belongs to the individual is still $1/k$. For example, the probability of knowing Freda Shields's ID is still 1/2. □

To illustrate further that k-anonymity does not deal with the attribution disclosure, we consider the following example.

Example 2.2.3. Assume that X contains the names, the gender, the DOB and the income (SA) of all graduate students in the Department of Mathematics and Statistics, uOttawa and Y is obtained via k-anonymization as in the previous example. In particular, we have 100 graduate students, and one equivalence class will be of the form "Male, born in 1995" and will likely consist of several students. So, from the technical point of view, the dataset Y will be e.g. 5-anonymized. However, it is almost certain that all the students in this equivalence class will have the same income. Hence, even after k-anonymization we did not achieve anything from the point of view of data privacy with respect to the income. □

In Example 2.2.2, we assumed that the attacker knows that Alan Smith is in the original database X. It is possible instead, that the attacker knows only that Alan Smith is in a bigger dataset Z, while X is obtained as a subset of Z. Then, as before, Y is obtained via k-anonymization of X. The attacker is then not sure if Alan Smith is in the released database Y. Hence, re-identification probability might be much smaller than $1/k$. This is addressed below, based on [9].

Problem Setup: We have three datasets:

- **The identification dataset** Z of size N. It is treated as an *external file or population data*, indexed by $U = \{1, \dots, N\}$. It contains Explicit Identifiers (EIs) and Quasi-Identifiers (QIs), but does not contain Sensitive Attributes. The entries of the database are denoted by $Z_1, \ldots, Z_N$. This dataset is public.

- **The dataset** X of size n (where $n \leq N$). It is treated as *sample or original data*, indexed by $S = \{1, \ldots, n\}$. It contains EIs, QIs, and Sensitive Attributes. Although X is treated as a sample from Z, formally, records in X cannot be obtained by sampling records from Z because the latter is missing the target variables. The entries of the database are denoted by $X_1, \ldots, X_n$, with the idea being to sample individual IDs from the population and obtain their QIs and sensitive attributes.

- **The anonymized dataset** Y of size m (which may not equal n). It is obtained from X via an anonymization mechanism, such as *suppression or generalization*, to achieve k-anonymity. It is also treated as "released data," indexed by $S' \subseteq S$. The entries of the database are denoted by Y_i, $i \in S'$. This dataset is available to the attacker.

Example 2.2.4. This example is based on [9]; see Example 2.2.2. In this example, an intruder has access to the identification database (external file) denoted by Z. The intruder then attempts re-identification by matching an arbitrary record in Z against the records in the published database Y on year of birth and gender. In our example, once an arbitrary individual is re-identified, the intruder will have that individual's ID number in the anonymized data and hence its test result. The database Y is obtained through 2-anonymization process from the original database X.

For this example,

- The dataset Z consists of 31 people and contains their names, gender and DOB.

- The database X contains $n = 14$ people with their names, gender and date of birth (EIs and QIs) as well as medical test results (SA).

- The dataset Y has been obtained from X via 2-anonymization. The names and ID are removed, the gender is kept, while the DOB is replaced with a range of dates. The test results are kept. The ID is kept in the example for demonstration purpose.

- We have $J = 5$ equivalence classes that correspond to "Male, [1950-1959]", "Male, [1960-1969]" etc.

The main difference between the disclosure risk (identity disclosure or attribution disclosure) in this setup and Example 2.2.2 is as follows: Example 2.2.2 assumes the attacker knows the individual is in the original dataset, focusing on the anonymization process only between X and the anonymized dataset Y. The setup of Example 2.2.4 considers an additional dataset Z, representing a larger population, which introduces uncertainty about whether an individual is actually included in X. $\square$

In summary,

- **k-anonymity measures identity disclosure, not attribution disclosure**. The probability of correctly identifying an individual is at most $\frac{1}{k}$, assuming that an adversary does not have access to additional external information that could be used to narrow down the individual's identity further.
- As the value of k increases, each individual's data becomes hidden among a larger group of people with similar quasi-identifier attributes. This increase in k directly enhances privacy by reducing the likelihood of re-identification based on these attributes.

2.3 Beyond k-anonymity: ℓ-diversity

ℓ-diversity. This concept was introduced in [19]. The ℓ-diversity is supposed to address the limitations of k-anonymity mentioned in the previous section. **This approach not only protects against identity disclosure, but also against attribution disclosure.** This is achieved by ensuring that each equivalence class in the dataset contains a diverse set of sensitive attributes, which helps prevent any potential attribute disclosure. The number ℓ can be interpreted as the minimum number of distinct, well-represented sensitive attribute values required within each equivalence class to rule out certain inferences about an individual's attributes.

ℓ-diversity is a measure of attribution disclosure which is formally defined as:

Definition 2.3.1 (ℓ-diversity)**.** *An equivalence class is said to have ℓ-diversity if there are at least ℓ well-represented values for the sensitive attribute. The table is said to have ℓ-diversity if every equivalence class of the table has ℓ-diversity.*

Note that ℓ-diversity is similar in concept to k-anonymity but targets a different aspect of privacy. While k-anonymity focuses on the indistinguishability of individuals based on *Quasi-Identifiers*, ℓ-diversity ensures diversity in *Sensitive Attributes* within these groups. This distinction is crucial for preventing attribute-based inferences.

Problem Setup:

- To calculate the disclosure risk measure, we only need the released dataset, either X or Y, depending on which dataset has been made available to the public or potential attackers.

Example 2.3.2. Assume the data holder releases the dataset Y from Example 2.2.2. For each equivalence class we have:

- Female, 1970-1979: Distinct Test Results = {-ve} $\rightarrow$ ℓ-diversity = 1,
- Female, 1960-1969: Distinct Test Results = {-ve, Zero} $\rightarrow$ ℓ-diversity = 2,
- Male, 1950-1959: Distinct Test Results = {+ve, -ve} $\rightarrow$ ℓ-diversity = 2,
- Male, 1960-1969: Distinct Test Results = {-ve} $\rightarrow$ ℓ-diversity = 1,
- Male, 1970-1979: Distinct Test Results = {-ve} $\rightarrow$ ℓ-diversity = 1.

The overall ℓ-diversity of the dataset is determined by the lowest diversity value in any equivalence class, which is 1. This indicates that the dataset does not adequately protect against attribute disclosure for most equivalence classes. We mentioned this already before.
In order to achieve ℓ-diversity, one can create additional records. For example, consider the following record.

11	Female	1960-1969	-ve
11′	Female	1960-1969	-ve
7	Female	1960-1969	Zero

In the equivalence class Female, 1960-1969, there are two possible outcomes (-ve and Zero). Without additional information, an attacker would guess -ve or Zero with equal probability, so the probability of correctly guessing the sensitive attribute for an individual from this equivalence class will be $\frac{1}{2}$. However, when the sensitive attribute is not unique, the probabilistic interpretation of ℓ-diversity is lost. Assuming there are three people in the equivalence class Female, 1960-1969, there are still two possible outcomes (-ve and Zero), that is: The probability of correct guessing attributes for individual with ID 11 and 11′ will be 2/3 instead of 1/2. □

Example 2.3.3. In some extreme cases, ℓ-diversity may fail to adequately protect privacy. For example, consider an equivalence class where there are 99 records with a positive result and only 1 record with a negative result. In this scenario, ℓ-diversity does not guarantee privacy for the individual associated with the negative result. Furthermore, it also cannot handle the continuous target and key variable. For example, assume we have another Sensitive Attribute, the weight: Here, the weights 55.1, 55.3,

11	Female	1960-1969	-ve	55.1
11′	Female	1960-1969	-ve	55.3
7	Female	1960-1969	Zero	55.4

and 55.4 differ slightly only, the disclosure metric would still return 3-diversity because each weight is considered distinct. This can give a misleading measure of privacy protection, as the slight variations in weight can not mask individual attribute effectively. □

In summary,

- **ℓ-diversity measures both identity and attribution disclosure** by ensuring that each equivalence class contains at least ℓ well-represented, distinct values for sensitive attributes.
- As ℓ increases, the diversity within each equivalence class also grows, making it more challenging for an attacker to make accurate assumptions or inferences about any individual's sensitive attributes. Consequently, a higher ℓ value enhances privacy protection.

2.4 Beyond k-anonymity: t-closeness

t-closeness. This concept was introduced in [17]. t-closeness is a further generalization of k-anonymity and ℓ-diversity. It takes into account the distribution of the Sensitive Attribute values.

Definition 2.4.1 (t-closeness)**.** *An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute across the entire table does not exceed a threshold t. Mathematically, it is defined as:*

$$d\{P_j, Q\} \leq t \ \forall \ j;$$

where P_j is the distribution within the equivalence class $[j]$, Q is the distribution of the released data, and d represents a distance function between probability distributions.

A table is said to have t-closeness if all its equivalence classes achieve t-closeness. This measure assesses the similarity between the distribution of a sensitive attribute within any given equivalence class and the overall distribution in the dataset. A small distance indicates that the sensitive attribute's distribution in the class does not significantly differ from its overall distribution in the dataset, thereby making it difficult to predict the sensitive value based on non-sensitive attributes within any given class.

Example 2.4.2. Continuing with Example 2.2.2, we show calculation of the t-closeness for each equivalence class by determining the empirical distribution of the Sensitive Attribute, the test result, from the released data. We observed 2 positive, 8 negative, and 1 zero result in dataset Y. Thus, the probabilities of test result for the dataset are: $\frac{2}{11}$ for positive, $\frac{8}{11}$ for negative and $\frac{1}{11}$ for zero.

For the equivalence class of Males from 1950-1959, we observed:

- 2 Positive and 1 Negative test result, giving probabilities of $\frac{2}{3}$ for Positive, $\frac{1}{3}$ for Negative and 0 for Zero.

The KL-divergence for this equivalence class, using the overall distribution, is calculated as:

$$d_{KL} = \frac{2}{3}\log\left(\frac{2/3}{2/11}\right) + \frac{1}{3}\log\left(\frac{1/3}{8/11}\right) = 0.606.$$

Similarly, for other equivalence classes, we calculate:

- For Males 1960-1969, we have only negative results: $d_{KL} = 1\log\left(\frac{1}{8/11}\right) = 0.318$.
- For Males 1970-1979, we have only negative results: $d_{KL} = 1\log\left(\frac{1}{8/11}\right) = 0.318$.
- For Females 1950-1959, we have only negative results: $d_{KL} = 1\log\left(\frac{1}{8/11}\right) = 0.318$.
- For Females 1960-1969, we have one negative results and one zero result: $d_{KL} = \frac{1}{2}\log\left(\frac{1/2}{8/11}\right) + \frac{1}{2}\log\left(\frac{1/2}{1/11}\right) = 0.752$.

Thus, we choose the largest D_{KL} and the released database Y has $t = 0.752$ closeness. What does it mean? We cannot really interpret this number on its own; however, we can consider another database and calculate its t-closeness. This will give us a comparison of the databases in terms of privacy.
Now, we modify the dataset Y as follows. Intuitively, this table is more private. We

Table 2.5: Table Y

	QI		SA
ID	Gender	Decade of Birth	Test Result
13	Female	1960-1979	-ve
6	Female	1960-1979	-ve
11	Female	1960-1979	-ve
7	Female	1960-1979	+ve
12	Male	1950-1979	+ve
1	Male	1950-1979	+ve
4	Male	1950-1979	-ve
2	Male	1950-1979	-ve
14	Male	1950-1979	-ve
9	Male	1950-1979	-ve
10	Male	1950-1979	-ve

recalculated the KL-divergence for the new equivalence classes to determine the new t-closeness. The overall distribution of test results remains the same: $+ve = \frac{2}{11}$, $-ve = \frac{8}{11}$, $Zero = \frac{1}{11}$.

- For the combined male class, we have $d_{KL} = 0.116$.

- For the combined female class, we have $d_{KL} = 0.276$.

Thus, we choose the largest D_{KL} and the released database Y has $t = 0.276$ closeness. Notice that compared to Table 2.4, the value of t is smaller, indicating better privacy. $\square$

We note that t-closeness equal to zero means "perfect privacy" - all equivalence classes follow the same distribution as the population distribution. A larger t allows for a greater discrepancy between the attribute distributions in each equivalence class and the entire dataset. Thus, a smaller value of t improves privacy by ensuring that the sensitive attributes of any group cannot be distinguished significantly from those of the overall population.

2.5 Differential privacy

This concept was introduced in [7] and became extremely popular in the last few years. It has a completely different interpretation, compared to the privacy measures discussed above. Differential privacy ensures that the addition or removal of an individual's data does not significantly change the results of a statistics. The goal is to protect the privacy of individual data while allowing the analysis of aggregated data.

In what follows, we denote:

- D and D' as any two datasets that differ by at most one element, often referred to as "neighboring datasets." This difference could be the addition or removal of a single record (an individual's data).

- Epsilon (ϵ) as a non-negative real number that quantifies the level of privacy protection provided by the mechanism. Smaller values of ϵ indicate stronger privacy guarantees. It essentially controls the permissible variation in output probabilities when comparing two neighboring datasets.

- Mechanism ($\mathcal{M}$) as the randomized mechanism or algorithm used. This mechanism takes a data set as input and outputs a result that has been added to the noise or has been transformed in some other way to protect privacy.

- Output set (S) as any measurable subset of possible outputs of the mechanism $\mathcal{M}$. These sets contain all possible outcomes that the mechanism can produce.

- The function f with values in $\mathbb{R}^k$ is a query function, which is used to calculate specific information or statistics from the dataset, such as sums, averages, or more complex outputs.

- Δf as the sensitivity of the function f, defined as:

  $$\Delta f = \sup_{D,D'} \|f(D) - f(D')\|,$$

  where $\|\cdot\|$ is a norm on $\mathbb{R}^k$. Sensitivity quantifies the maximum difference in the output of a function when applied to two neighboring datasets. The amount of noise added is directly proportional to the sensitivity, ensuring that we understand the worst-case impact that a single individual's data can have on the function's output, thus providing a bound for the likelihood ratio.

Definition 2.5.1. *A randomization mechanism $\mathcal{M}$ satisfies ϵ-differential privacy if for all datasets D and D' and for all measurable output sets S, the following inequality holds:*

$$\sup_S \left| \log\left(\frac{\mathbb{P}[\mathcal{M}(D) \in S]}{\mathbb{P}[\mathcal{M}(D') \in S]} \right) \right| \leq \epsilon.$$

The logarithm converts the probability ratio into a quantifiable measure of information in information theory, typically measured in "bits". This measure helps to assess how much information about an individual is revealed by the output of a randomized mechanism. When taking the logarithm of a probability ratio, the result can be interpreted as the amount of information gained or lost, making it easier to assess privacy risks. The definition of differential privacy ensures that the likelihood ratio is always bounded between e^{ϵ} and $e^{-\epsilon}$, indicating that the maximum change in information, due to the presence or absence of a single individual's data, is tightly controlled.

The Laplace mechanism is a fundamental tool for achieving differential privacy using an additive noise mechanism. It adds noise to the output of a function to protect individual privacy. The noise comes from the Laplace distribution, which is characterized by a scale parameter determined by the desired level of privacy and the sensitivity of the function f. The Laplace mechanism is a postprocessing algorithm, meaning that any function applied to the output of a differentially private mechanism remains differentially private.

Theorem 2.5.2. *Let D be a database and let f be a real-valued query. Define the Laplace mechanism $\mathcal{M}_L$ as*

$$\mathcal{M}_L(D) = f(D) + Lap(0, \lambda)$$

where $Lap(0, \lambda)$ is the Laplace distribution with mean 0 and scale parameter $\lambda = \frac{\Delta f}{\epsilon}$. Then, the Laplace mechanism satisfies differential privacy.

Proof. In what follows, we will write $g(x; Y)$ for the density of the random variable Y at the point x. Let D' be a neighbouring database. We have

$$\frac{g(x; \mathcal{M}_L(D))}{g(x, \mathcal{M}_L(D'))} = \frac{\exp\left(-\frac{|x-f(D)|}{\lambda}\right)}{\exp\left(-\frac{|x-f(D')|}{\lambda}\right)} = \exp\left(\frac{|x - f(D')| - |x - f(D)|}{\lambda}\right),$$

by the triangle inequality:

$$|x - f(D')| - |x - f(D)| \leq |f(D) - f(D')|.$$

Since $|f(D) - f(D')| \leq \Delta f$, we have:

$$\frac{g(x; \mathcal{M}_L(D))}{g(x, \mathcal{M}_L(D'))} \leq \exp\left(\frac{\Delta f}{\lambda}\right),$$

Given that $\lambda = \frac{\Delta f}{\epsilon}$, the expression simplifies to:

$$\frac{g(x; \mathcal{M}_L(D))}{g(x, \mathcal{M}_L(D'))} \leq \exp(\epsilon).$$

Therefore, the Laplace mechanism satisfies ϵ-differential privacy, as the ratio of probabilities is bounded by e^{ϵ}. □

In summary, lower values of ϵ imply more privacy protections at the cost of adding more noise to the data, which can affect the data's utility for detailed analysis.

Local Differential Privacy (LDP) modifies the traditional concept of differential privacy by applying privacy-preserving randomization at the individual data source level.

Definition 2.5.3. *A local randomization mechanism $\mathcal{M}$ satisfies ϵ-local differential privacy if, for any data entries x and x', and for all measurable output sets S, the inequality below holds:*

$$\sup_S \left| \log\left(\frac{\mathbb{P}[\mathcal{M}(x) \in S]}{\mathbb{P}[\mathcal{M}(x') \in S]}\right) \right| \leq \epsilon.$$

Notice that local differential privacy is a sqecial case of differential privacy where the qurey function $f(D)$ is defined as a identity function applied to the individual's data point. In LDP, when we refer to neighboring datasets D and D', we consider two different possible values that an individual's data point can take, x and x'.

Theorem 2.5.4. *Let $\mathcal{M}_L$ be a mechanism defined by*

$$\mathcal{M}_L(x) = x + Lap\left(0, \frac{\Delta Id}{\epsilon}\right),$$

where ΔId is the sensitivity of the identity query, representing the maximum difference between two arbitrary data points. Then, $\mathcal{M}_L$ satisfies ϵ-local differential privacy.

Proof. For any two data points x and x' and output s, the probability ratio is given by:

$$\frac{g(s; \mathcal{M}_L(x))}{g(s; \mathcal{M}_L(x'))} = \exp\left(\frac{|s - x'| - |s - x|}{1/\epsilon}\right).$$

Using the triangle inequality, we have:

$$|s - x'| - |s - x| \leq |x' - x|.$$

Given that the maximum change (sensitivity) between any two data points x and x' is ΔId, the exponent in the probability ratio can at most be ϵ, thus:

$$\frac{g(s; \mathcal{M}_L(x))}{g(s; \mathcal{M}_L(x'))} \leq \exp(\epsilon).$$

□

Definition 2.5.5. *A randomization mechanism $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy if for all datasets D and D' that differ on at most one element, and for all output subsets S, the following holds:*

$$\mathbb{P}[\mathcal{M}(D) \in S] \leq e^{\epsilon}\mathbb{P}[\mathcal{M}(D') \in S] + \delta$$

(ϵ, δ)-Differential Privacy: Unlike ϵ-differential privacy, (ϵ, δ)-differential privacy introduces a small probability δ, which accounts for the event that ϵ-differential privacy may not be strictly held.

Theorem 2.5.6. *Let D be a database. Let $f : \mathbb{R}^n \to \mathbb{R}^k$. The Gaussian mechanism is defined as*

$$\mathcal{M}_G(D) = f(D) + (Y_1, \ldots, Y_k),$$

where the Y_i are independent $\mathcal{N}(0, \sigma^2)$ random variables with $\sigma^2 = 2\ln(\frac{1.25}{\delta})\frac{\Delta^2}{\epsilon^2}$.

Proof. Without loss of generality, assume that $k = 1$. Define the privacy loss $L(x)$ as

$$L(x) = \log\left(\frac{g(x; \mathcal{M}_G(D))}{g(x; \mathcal{M}_G(D'))}\right).$$

Let $f(D') = f(D) + v$. Note that v depends on the databases D and D', however, $|v| < \Delta f$, the sensivity of the query f. Then

$$\begin{aligned} L(x) &= \log\left(\frac{\exp\left(-\frac{(x-f(D'))^2}{2\sigma^2}\right)}{\exp\left(-\frac{(x-f(D))^2}{2\sigma^2}\right)}\right) \\ &= \frac{-(x - f(D) - v)^2 + (x - f(D))^2}{2\sigma^2} \\ &= \frac{2(x - f(D))v - v^2}{2\sigma^2} \\ &= \frac{2\eta v - v^2}{2\sigma^2}, \end{aligned}$$

where $\eta = x - f(D)$. Here, x is an outcome of the randomization algorithm. Thus, η is a realization of the normal random variable Y_1. Then

$$\mathbb{P}\left(\frac{Y_1 v}{\sigma^2} - \frac{v^2}{2\sigma^2} \geq \epsilon\right) = \mathbb{P}\left(\frac{Y_1 v}{\sigma^2} \geq \epsilon + \frac{v^2}{2\sigma^2}\right).$$

Define $Z = Y_1/\sigma$, therefore $Z \sim \mathcal{N}(0,1)$, hence we need to evaluate

$$\mathbb{P}\left(Z \geq \frac{\epsilon\sigma - \frac{v^2}{2\sigma}}{v}\right).$$

Recall that

$$\sigma = \frac{\Delta f}{\epsilon}\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}, \quad |v| \leq \Delta f.$$

Then

$$\begin{aligned}\mathbb{P}\left(Z \geq \frac{\epsilon\sigma - \frac{v^2}{2\sigma}}{v}\right) &\leq \mathbb{P}\left(Z \geq \sqrt{2\ln\left(\frac{1.25}{\delta}\right)} - \frac{\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}}{2}\right)\\ &\leq \mathbb{P}\left(Z \geq \frac{\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}}{2}\right).\end{aligned}$$

Given the properties of the Gaussian distribution:

$$\mathbb{P}(Z \geq z) \approx 1 - \Phi(z).$$

Thus:

$$\mathbb{P}\left(Z \geq \frac{\epsilon\sigma - \frac{v^2}{2\sigma}}{v}\right) = \mathbb{P}\left(Z \geq \frac{\sqrt{2\ln\left(\frac{1.25}{\delta}\right)}}{2}\right) \leq \delta.$$

Thus the Gaussian mechanism satisfies (ϵ, δ)-differential privacy. □

2.6 Correct attribution probability (CAP)

CAP, or Correct Attribution Probability, serves as a metric to measure the risk of disclosing sensitive information or target value through an inference or linking attack, assuming that some values in real data are public knowledge. An attacker might combine this known information with released (often anonymized) data, to make a prediction or guess about other sensitive values. CAP metric evaluates the difficulty an attacker faces in correctly guessing the sensitive information.

We start with the basic definition of CAP scores based on [12] and [32]; see Section 2.6.2. We provide several examples and the key finding is that population-level CAP scores, as defined in the aforementioned papers, may not be interpretable. In particular, these scores cannot be interpreted as probabilities (as originally intended in those papers). Furthermore, the traditional CAP is not adequate for continuous data. To address these issues, we propose modified CAP scores based on equivalence classes (Section 2.6.3). This way we achieve two goals: we are able to handle continuous data while preserving CAP scores interpretation in terms of probability measures.

Furthermore, we analysed a connection between anonymization through noise addition (as in differential privacy) and CAP calculation. In principle, the more noise, the more privacy, and this should an effect on CAP calculation. We will illustrate that this connection is not so obvious; see Section 2.6.6. Likewise, we analyse how the choice of equivalence classes affects calculations of CAP scores, and hence their interpretations in terms of privacy. We recognize similar issues as bandwidth choice in nonparametric estimation. See Section 2.6.7.

In summary, the original CAP scores proposed in the literature have a limited applicability and may lack interpretation in the context of privacy. We proposed CAP scores based on equivalence classes. This proposal solves some problems, but at the same time introduces another set of issues.

2.6.1 Set up and terminology

- We have two datasets: the original and the anonymized one.
- Both datasets include key (nonsensitive) variables and target (sensitive) variables. In the terminology introduced before, key variables correspond to Quasi-Identifiers, while target variables correspond to Sensitive Attributes.
- When the anonymized key matches the original key, we will talk about the **match**.
- When (anonymized key, anonymized target) is matched with (original key, original target), we will discuss the **correct match**.
- $K_{O,j}$ is the key value for the jth individual in the original dataset.
- K_o is a random variable that represents the key values of individuals in the original dataset.
- $T_{O,j}$ is the target value for the jth individual in the original dataset.
- T_o is a random variable that represents the target values of individuals in the original dataset.

- $K_{A,j}$ is the key value for the jth individual in the anonymized dataset.
- K_a is a random variable that represents the key values of individuals in the anonymized dataset.
- $T_{A,j}$ is the target value for the jth individual in the anonymized dataset.
- T_a is a random variable that represents the target values of individuals in the anonymized dataset.
- n_O is the number of observations in the original dataset.
- n_A is the number of observations in the anonymized dataset.
- $Id_{O,j}$ is the identifier for the jth individual in the original database, while $Id_{A,i}$ is just an indicator for an ith entry in the anonymized dataset. It is important to note that $Id_{A,i}$ does not necessarily refer to the identifier of this individual in the original data set.
- When the anonymized dataset is released, the attackers have an information about the pair of variables T_a, K_a and we assume that the attacker also has an access to an external dataset that contains both the variable K_o and explicit identifiers EI.
- It is also assumed that the keys K_o and the explicit identifier pair (EI) in the external data set are exactly the same as the pairs in the original dataset; note that this assumption will not always hold in practice.

2.6.2 Original CAP

There are two types of scores that we can calculate. The first is the **record level score**, which offers information about an individual's or equivalence class risk of disclosure. The second is the **population level score**, which provides an overall measure of risk for the entire dataset.

CAP record level score based on the original dataset:

$$\mathrm{CAP}_{O,j} = \frac{\sum_{i=1}^{n_O} 1\{T_{O,i} = T_{O,j}, K_{O,i} = K_{O,j}\}}{\sum_{i=1}^{n_O} 1\{K_{O,i} = K_{O,j}\}} \ , \quad j = 1, \ldots, n_O \ . \tag{2.2}$$

This formula, in principle, approximates:

$$\mathbb{P}(T_o = T_{O,j} \mid K_o = K_{O,j}) \ ,$$

the conditional probability that a randomly selected individual from the original database has the particular attribute $(T_{O,j}, K_{O,j})$, given the randomly selected individual from the original has the particular key attribute $K_{O,j}$. This formula does not compare the anonymized dataset with the original one; it serves as a baseline.

CAP population level score for the original dataset:

$$\text{CAP}_{O,\text{score}} = \frac{1}{n_O} \sum_{j=1}^{n_O} \text{CAP}_{O,j} \ . \tag{2.3}$$

This score does not have a proper interpretation, which will be explained later.

CAP records level score based for anonymized dataset:

$$\text{CAP}_{A,j} = \frac{\sum_{i=1}^{n_A} 1\{T_{A,i} = T_{O,j}, K_{A,i} = K_{O,j}\}}{\sum_{i=1}^{n_A} 1\{K_{A,i} = K_{O,j}\}} \ , \quad j = 1, \ldots, n_A \ . \tag{2.4}$$

This formula, in principle, is an approximation to

$$\mathbb{P}(T_a = T_{O,j} \mid K_a = K_{O,j}) \ ,$$

the probability that a randomly selected individual from the anonymized database has the particular attribute $(T_{O,j}, K_{O,j})$ given that this individual has the particular key attribute $K_{O,j}$.

Intuitively, the bigger value of the CAP, the more similar the original and the anonymized database are, the less privacy.

CAP dataset score for anonymized dataset:

$$\text{CAP}_{A,\text{score}} = \frac{1}{n_A} \sum_{j=1}^{n_A} \text{CAP}_{A,j} \ . \tag{2.5}$$

In principle, if the $\text{CAP}_{A,\text{score}}$ is smaller than $\text{CAP}_{O,\text{score}}$ then the anonymized database has some level of privacy. However, does not always has such interpretation, as we will explain later.

Example 2.6.1. In this example, we demonstrate the calculation for the scores mentioned above: the dataset contains one key variable (gender) and one target variable (test result), both of which are categorical variables. We present this example to illustrate the calculation steps.

- The original database X contains $n_O = 4$ records;
- j correspond to the person in original data with $Id_{O,j}$;
- i correspond to the person in anonymized data with $Id_{A,i}$;
- $K_{O,j}$ can take value (M, F);

Table 2.6: X

Id$_O$	original key, Gender	original target, Test Result
1	M	P
2	M	P
3	F	P
4	F	N

Table 2.7: Z

Id$_O$	original key, Gender
1	M
2	M
3	F
4	F

Table 2.8: Y

Id_A	anonymized key, Gender	anonymized target, Test Result
1	M	P
2	F	P
3	F	N
4	F	N

- $T_{O,j}$ can take value (P, N).

The data holder can see the original data X, Table 2.6, and Y is the anonymized dataset. An adversary can see table Z, Table 2.7, and Y Table 2.8. Note that Id_A does not correspond to any explicit identifier (EI) in the original dataset; it is used here solely for demonstration purposes.

Calculations for CAP$_O$: We first provide calculation of CAP$_{O,j}$, we use formula (2.2), the jth individual score for original. Note that for CAP$_{O,j}$ individual score, we only need original dataset for calculation:

- $j = 1$: To calculate the denominator of formula (2.2), here $K_{O,1} = M$ and we have **two** key entries in the original dataset that match M. To calculate the numerator of formula (2.2), $(K_{O,1}, T_{O,1}) = (M, P)$ and we have **two** entries in the original dataset that match this pair. Hence, CAP$_{O,j=1} = 1$. This number has the proper interpretation: given that the original key is M, there is a 100% chance that the original target will be P.

- Similarly, for $j = 2$, CAP$_{O,j=2} = 1$.

- $j = 3$: To calculate the denominator of formula (2.2), here $K_{O,3} = F$ and we have **two** key entries in the original dataset that match F. To calculate the numerator of formula (2.2), $(K_{O,3}, T_{O3}) = (F, P)$ and we have **one** entry in the original dataset that matches this pair. Hence, $\text{CAP}_{O,j=3} = 1/2$. This number has the proper interpretation: given that the original key is F, there is a 50% chance that the original target will be P.

- Similarly, for $j = 4$, $\text{CAP}_{O,j=4} = 1/2$.

Calculations for CAP_A: We can calculate $\text{CAP}_{A,j}$, the j-th individual score for anonymized data, using formula (2.4):

- $j = 1$: To calculate the denominator of formula (2.4), here $K_{O,1} = M$ and we have **one** key entry in the anonymized dataset that matches M. To calculate the numerator of formula (2.4), $(K_{O,1}, T_{O,1}) = (M, P)$ and we have **one** entry in the anonymized dataset that matches this pair. Hence, $\text{CAP}_{A,j=1} = 1$. This number has the proper interpretation: given that the original key is M, there is a 100% chance that the anonymized target will be P.

- Similarly, for $j = 2$, we have $(K_{O,2}, T_{O,2}) = (M, P)$, hence $\text{CAP}_{A,j=2} = 1$.

- $j = 3$: To calculate the denominator of formula (2.4), here $K_{O,3} = F$ and we have **three** key entries in the original dataset that match F. To calculate the numerator of formula (2.4), $(K_{O,3}, T_{O,3}) = (F, P)$ and we have **one** entry in the original dataset that matches this pair. Hence, $\text{CAP}_{A,j=3} = 1/3$. This number has the proper interpretation: given that the original key is F, there is a 1/3 chance that the original target will be P.

- $j = 4$: Here $K_{O,4} = F$ and we have **three** key entries in the original dataset that match F. To calculate the numerator of formula (2.4), $(K_{O,4}, T_{O,4}) = (F, N)$ and we have **two** entries in the original dataset that match this pair. Hence, $\text{CAP}_{A,j=4} = 2/3$. This number has the proper interpretation: given that the original key is F, there is a 2/3 chance that the original target is N.

So, the record level CAP scores have the proper interpretation of the conditional probability and give us some information about privacy. At the same time, both $\text{CAP}_{O,\text{score}}$ and $\text{CAP}_{A,\text{score}}$ are equal to 3/4. **Hence, we cannot here get any interpretation of these scores in terms of privacy.** □

Example 2.6.2. In this example, we show calculation of CAP scores in a multivariate case. We extend the previous dataset, by adding another variable, age. We now have 2 key variable and 1 target variable, age is treated as the categorical variable. Notice that the target and the key are both unique in the original dataset. We will calculate CAP dataset scores, (2.3) and (2.5).

Similarly, only the data holder can see the original table X, Table 2.9. An adversary will see table Z; Table 2.10 the original data with sensitive attributes removed, and Y 2.11 the anonymized dataset.

Table 2.9: X

Id_O	original key_1 (Age)	original key_2 (Gender)	original target (Test Result)
1	18	M	A
2	19	M	B
3	20	M	C
4	21	M	D
5	18	F	E
6	19	F	F
7	20	F	G
8	21	F	H

Table 2.10: Z

Id_O	original key_1 (Age)	original key_2 (Gender)
1	18	M
2	19	M
3	20	M
4	21	M
5	18	F
6	19	F
7	20	F
8	21	F

Table 2.11: Y

Id_A	anonymized key_1 (Age)	anonymized key_2 (Gender)	anonymized target (Result)
1	18-21	M	A
2	18-21	M	B
3	18-21	M	C
4	18-21	M	D
5	18-21	F	E
6	18-21	F	F
7	18-21	F	G
8	18-21	F	H

- To create a database Y (Table 2.11), we consider all entries in X and then apply 4-anonymization with respect to gender and age.
- $K_{O,j}$ is the key value for the j-th individual in the original dataset. $K_{O,j}$ can take values $\{(18, M), (19, M), \ldots\}$.
- $T_{O,j}$ is the target value for the j-th individual in the original dataset. $T_{O,j}$ can take values $\{A, B, C, D, E, F, G, H\}$.

To calculate $\text{CAP}_{O,\text{score}}$, we use the formula (2.4). In what follows, j stands for the j-th individual in the data. For example:

- For $j = 1$, $\text{CAP}_{O,j} = 1$. Indeed, here $K_{O,j} = (18, M)$ and we have **one** key match in the original dataset. Also, $(T_{O,j}, K_{O,j}) = (18, M, A)$ and we have **one** pair in the original dataset that matches the original pair.
- Similarly, $\text{CAP}_{O,j} = 1, \forall j$.
- Hence, using (2.3), $\text{CAP}_{O,\text{score}} = 1$.
- Thus, the 1 value for the CAP score can be interpreted as a "perfect match," since we are comparing the original dataset with itself.

To calculate $\text{CAP}_{A,\text{score}}$, we use the formula (2.4). For example:

- $\text{CAP}_{A,1} = 1/4$. Indeed, here $K_{O,j} = (18, M)$ and we have **four** key matches in the anonymized dataset. Also, $(T_{O,j}, K_{O,j}) = (18, M, A)$ and we have **one** pair in the anonymized dataset that matches the original pair.
- Similarly, $\text{CAP}_{A,j} = 1/4 \ \forall \ j$.
- Hence, using (2.5), $\text{CAP}_{A,\text{score}} = 1/4$.

Hence, the CAP score for the anonymized data set is much lower than for the original data set. In this example, the $\text{CAP}_{A,\text{score}}$ provides some interpretation in terms of privacy. □

Issues with CAP scores

CAP dataset scores are easy to calculate, but often lack the proper interpretation (in principle, the lower CAP score, the more privacy). Specifically, the scores of the CAP dataset, calculated by averaging the sum of each CAP at record level, cannot be directly interpreted as probabilities. Indeed, the CAP score calculations $\text{CAP}_{O,\text{score}} = \frac{1}{n_O}\sum_{j=1}^{n_O} \text{CAP}_{O,j}$ are intended to represent the probability of accurate predictions based on the matched keys between the original and anonymized datasets. However, if these calculations do not consider equivalence classes, their interpretation as probabilities becomes incorrect.

- In the provided example, both $\text{CAP}_{O,j=1}$ and $\text{CAP}_{O,j=2}$ attempt to measure the conditional probability that an individual in the original dataset, identified by a certain key, satisfies a target attribute. However, these are not isolated instances; they encompass multiple occurrences within the same key equivalence class (the same key, different targets).

- Interpreting these scores as distinct probabilities without recognizing that they pertain to repeated occurrences within the same equivalence class leads to overcounting of the probability.

- For a simple example, the probability $\mathbb{P}(T_o = T_{O,j})$, which expresses the likelihood of observing the target value $T_{O,j}$ for any individual chosen at random from the original dataset, is given by:
$$\mathbb{P}(T_o = T_{O,j}) = \frac{\sum_{i=1}^{n_O} 1\{T_{O,i} = T_{O,j}\}}{n_O}.$$

- The indicator sum:
$$I := \sum_{j=1}^{n_O} \mathbb{P}(T_o = T_{O,j})$$
aggregates the probabilities for each distinct target value appearing in the dataset. If I is the probability, it should be less than or equal to 1. However, due to multiplicity—where several entries share the same target value, I can exceed 1. This overcount occurs because the sum treats repeated values within the same equivalence class as independent probabilities. This miscalculation demonstrates that simply adding each j without adjusting for equivalence classes loses the interpretation of probability; thus, the equation of the CAP score
$$\text{CAP}_{O,\text{score}} = \frac{1}{n_O} \sum_{j=1}^{n_O} \text{CAP}_{O,j}$$
cannot be interpreted as a probability.

This issue highlights the importance of considering the existence of equivalence classes when calculating CAP scores. We will discuss it thoroughly in Section 2.6.3.

Furthermore, the CAP score, as defined, has limitations when applied to continuous data. Even a minor change in continuous data is captured by the CAP score as a difference value, which can be misleading and underestimate the risk of disclosure.

Example 2.6.3. Consider the following original dataset: In this case, the CAP score for the anonymized dataset would return 0 (indicating possibly the perfect privacy). This result occurs because the CAP calculation treats any difference, no matter how small, as significant. However, such minor differences in practice do not affect privacy. To address this limitation, it is more practical to consider equivalence class methods for CAP. These methods group similar continuous values into categories or bins. □

Table 2.12: Original and Anonymized Datasets

Id_O	Original key (Height)	Original target (Weight)
1	160	48
2	161	52
3	168	52
4	170	59

Id_A	Anonymized key (Height)	Anonymized target (Weight)
1	160	48.01
2	161	52.01
3	168	52.01
4	170	59.01

2.6.3 Record-level equivalence classes CAP

We aim to address two primary challenges: first, multiplicity, which will be resolved using equivalence classes, and second, handling of continuous data, also resolved through a generalized definition of equivalence classes, as outlined below.

Problem Setup: The problem setup considers a common scenario in linkage attacks where an attacker has potential access to two manipulated versions of an original dataset. The two datasets typically involved are:

1. A dataset from which explicit identifiers (EI) have been removed, along with the application of a possible randomization algorithm. This dataset, often treated as the anonymized or released data, is assumed to be accessible to the public.

2. A dataset where sensitive attributes are removed, but explicit identifiers (EI) and key variables are retained. This dataset is treated as external data and may be available through other sources, providing additional information to the attacker. While this is not the only possible scenario, considering this context is important for interpreting the original CAP as a probability of prediction. Under this assumption, we can also interpret the equation (2.8) below as the probability of a correct prediction for a given target key equivalence class match.

Equivalence Classes:

- $K_{[j_1]}$ represents the j_1th equivalence class for the key variable, indexed by $j_1 \in 1 \ldots J_1$, where J_1 indicates the total number of equivalence classes for the key variable.

- $T_{[j_2]}$ represents the j_2th equivalence class for the target variable, indexed by $j_2 \in 1 \ldots J_2$, with J_2 representing the total number of equivalence classes for the target variable.

It is assumed that the equivalence classes are the same for both original and anonymized dataset. As such, in the notation, we omit the A and O subscript.

Equivalence classes can be intervals or single numbers, which extend the concept of original CAP. Proper setup of these classes requires that $T_{A,j}$, the target values in the anonymized dataset, fall within the union of the target equivalence classes. Formally, this can be represented as:

$$T_{A,j} \in \bigcup_{j_2} T_{[j_2]}.$$

Additionally, the intersection of different equivalence classes for both keys and targets should be empty, ensuring that each class is mutually exclusive with each other:

$$\bigcap_{j_1 \neq j_2} K_{[j_1]} \cap K_{[j_2]} = \emptyset \quad \text{and} \quad \bigcap_{j_1 \neq j_2} T_{[j_1]} \cap T_{[j_2]} = \emptyset.$$

Linkage and Attack Prediction: The attacker typically employs a straightforward record linking technique, focusing on matching individuals from an external data set to those in an anonymized data set. In this approach, the attacker starts with each individual in the external dataset and attempts to find a corresponding **match** in the anonymized dataset, then the attacker tries to predict the target attributes of individuals from the external dataset based on the anonymized data.

- For a given $[j_1], [j_2]$, a **match** is recognized when one or multiple individuals j from the anonymized data satisfy $K_{A,j} \in K_{[j_1]}$, and there exists one or more $K_{O,i}$ within $K_{[j_1]}$. Again, we assume the external dataset contains original key-target attribute pairs K_o and T_o.

- The attacker's prediction of the sensitive attributes $T_{[j']}$ for the target value $T_{O,i}$ of the matching unit would then be one or more values from $T'_{[j_2]}$, where $T'_{[j_2]}$ are selected if for some j, $T_{A,j} \in T_{[j_2]}$. If multiple $T_{[j_2]}$ are selected, the probabilities of choosing a specific $T'_{[j_2]}$ depend on the problem setup and the level of data access, as explained later in Example 2.6.6.

- When $T_{O,j}$ is within the range of selected $T'_{[j_2]}$, it is considered as a **correct match**. The equation (2.8) quantifies the probability that an attacker can correctly predict the target value for individuals within a given key target equivalence class $[j_1], [j_2]$ given a match.

Formulas: Now, we propose the generalized CAP record level scores for both original and anonymized dataset.

CAP individual score based on the original dataset:

$$\text{CAP}_{O,[j_1],[j_2]} = \frac{\sum_{i=1}^{n_O} 1\{T_{O,i} \in T_{[j_2]}, K_{O,i} \in K_{[j_2]}\}}{\sum_{i=1}^{n_O} 1\{K_{O,i} \in K_{[j_2]}\}} \ . \tag{2.6}$$

This formula represents the conditional probability that a randomly selected pair "(original target, original key)" belongs to the particular equivalence class, given that the randomly selected "original key" belongs to that equivalence class. That is,

$$\mathbb{P}(T_o \in T_{[j_2]} \mid K_o \in K_{[j_1]}).$$

Assuming the data holder released the original data with the explicit identifiers (EI) removed but without applying anonymization techniques, and considering the same external dataset mentioned in the problem setup, this formula estimates the probability that an attacker can accurately predict the target value for individuals within the same target equivalence class $[j_2]$, who are grouped within the same key equivalence class $[j_1]$. $\mathrm{CAP}_{O,[j_1],[j_2]}$ equals 1 when there is exactly one individual in each key equivalence class, implying that for every match identified by an attacker, there is only one possible prediction available, which is also a correct match. However, if there are multiple individuals within the same key equivalence class who belong to different target equivalence classes, the attacker's prediction of the target class becomes random, depending on the distribution of target classes within that specific key equivalence class.

Our next goal is to introduce the CAP score for the anonymized data set. In the spirit of (2.6), the natural proposal would be

$$\frac{\sum_{i=1}^{n_A} 1\{T_{A,i} \in T_{[j_2]}, K_{A,i} \in K_{[j_1]}\}}{\sum_{i=1}^{n_A} 1\{K_{A,i} \in K_{[j_1]}\}}\,, \tag{2.7}$$

$$\forall\ [j_1] \in \{1 \ldots J_1\}\ s.t.\ \exists\ K_{O,i} \in K_{[j_1]}.$$

This formula represents the conditional probability that a randomly selected pair:

(anonymized target, anonymized key)

belongs to a particular equivalence class, given that a randomly selected "anonymized key" belongs to that equivalence class. However, this formula does not involve original data, which creates a problem in interpreting the probability of a correct prediction. This issue will be demonstrated in Example 2.6.4.

CAP individual score based on the anonymized dataset:

Let $[j_1]$ be an equivalence class for the key variable such that there exist i, i' such that $K_{O,i}, K_{A,i'} \in K_{[j_1]}$. Define then

$$\mathrm{CAP}_{A,[j_1],[j_2]} = 1\{(K_{O,i}, T_{O,i}) \in (K_{[j_1]}, T_{[j_2]}) \text{ for some } i\} \frac{\sum_{i=1}^{n_A} 1\{T_{A,i} \in T_{[j_2]}, K_{A,i} \in K_{[j_1]}\}}{\sum_{i=1}^{n_A} 1\{K_{A,i} \in K_{[j_1]}\}}. \tag{2.8}$$

If there are no $K_{O,i}, K_{A,i'}$ that belong to $K_{[j_1]}$, the CAP score above will not be defined.

Since the attacker will ignore any equivalence class that does not make a match from the external data (original with target removed), to calculate the CAP score, we only consider a subset of j_1 such that the equivalence class $K_{[j_1]}$ is not empty in terms of both external and anonymized data.
This formula represents the estimation of the conditional probability that a randomly selected pair (anonymized target, anonymized key) belongs to the particular equivalence class that also contains an original pair, given that a randomly selected anonymized key belongs to the particular equivalence class that also contains an original key:

$$1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\}\mathbb{P}(T_a \in T_{[j_2]} \mid K_a \in K_{[j_1]})$$

This number has the proper interpretation: given that both the anonymized and original target equivalence classes have attributes in $K_{[j_1]}$, there is a $\text{CAP}_{A,[j_1],[j_2]}$ probability that the anonymized pair will have an original value in $T_{[j_2]}$.

Example 2.6.4. In this example, we revisit the scenario from Example 2.6.1 with a modified data set to illustrate the issues with the initial CAP formula (2.7).

Table 2.13: X

Id_O	Original key (Gender)	Original target (Test Result)
1	M	P
2	M	P
3	M	P
4	M	P

Table 2.14: Z

Id_O	Original key (Gender)
1	M
2	M
3	M
4	M

Table 2.15: Y

Id_A	Anonymized key (Gender)	Anonymized target (Test Result)
1	M	P
2	M	N
3	F	P
4	F	N

- Only the data holder can see Table X, which contains the original data. The adversary, on the other hand, can only access Table Z, where the sensitive attributes have been removed from the original data. Table Z can be treated as an external file, and Y represents the anonymized dataset. Note that Id_A does not correspond to any explicit identifier (EI) in the original dataset; it is used here solely for demonstration purposes

- Since there are no females in the original dataset, an attacker cannot link specific female individuals from the anonymized dataset.
- Furthermore, since the original sensitive attributes do not include negative results, the attacker's prediction of a negative result will be considered incorrect.

Issue with Formula (2.7)**:**

- For $[j_1] = 1, [j_2] = 1$:
  - Denominator: $K_{[j_1]=1} = M$ has two matching entries in the anonymized dataset.
  - Numerator: $(K_{[j_1]=1}, T_{[j_2]=1}) = (M, P)$ has one matching entry in the anonymized dataset.
  - CAP Score: $\text{CAP}_{A,[j_1]=1,[j_2]=1} = \frac{1}{2}$.
- For $[j_1] = 1, [j_2] = 2$:
  - Denominator: $K_{[j_1]=1} = M$ has two matching entries in the anonymized dataset.
  - Numerator: $(K_{[j_1]=1}, T_{[j_2]=2}) = (M, N)$ has one matching entry in the anonymized dataset.
  - CAP Score: $\text{CAP}_{A,[j_1]=1,[j_2]=2} = \frac{1}{2}$.
  - Notice that this calculation is incorrect because the prediction for (M, N) will be considered incorrect since there is no original data with such attributes. Thus, we need to consider $1\{T_o \in T_{[j_2]}\}$ in equation (2.8).
- For $[j_1] = 2, [j_2] = 1$ and $[j_1] = 2, [j_2] = 2$:
  - Denominator: $K_{[j_1]=2} = F$ has zero matching entries in the anonymized dataset.
  - Notice that the denominator equals 0, which is why we only consider

$$\forall\ [j_1] \in \{j_1\}\ s.t.\ \exists\ i:\ K_{O,i} \in K_{[j_1]}.$$

  - The CAP score should not be calculated because the attacker will not make predictions for individuals with gender F, as there is no corresponding key in the original dataset.

$\square$

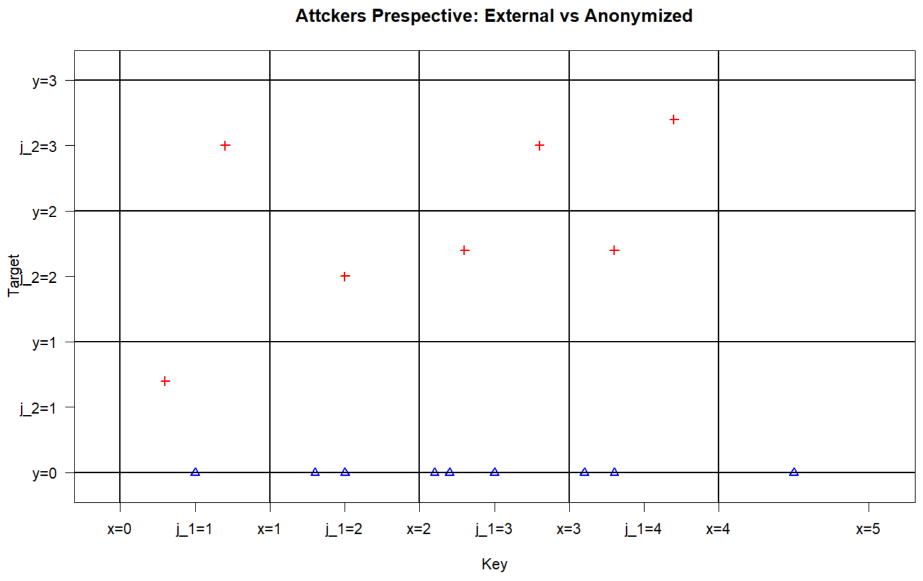


Figure 2.1: Attackers Perspective

2.6.4 Probability of a correct prediction for the target attributes

In this section, we show that $\text{CAP}_{A,[j_1],[j_2]}$ can be interpreted as the probability of correct prediction for the given $([j_1],[j_2])$ equivalence class under the assumption outlined in the problem setup.

Example 2.6.5. We first demonstrate how attackers make predictions for the target value based on the external and anonymized data set. In this example, we use the following datasets: The data holder can see the original data X, Table 2.16, and Y is the anonymized dataset. An adversary can see external dataset Z, Table 2.17, and Y Table 2.18. Note that Id_A does not correspond to any explicit identifier (EI) in the original dataset; it is used here solely for demonstration purposes. The equivalence classes for key K are defined as intervals $K_{[0,1)}$, $K_{[1,2)}$, $K_{[2,3)}$, and $K_{[3,4)}$, while the equivalence classes for target T are defined as $T_{[0,1)}$, $T_{[1,2)}$, and $T_{[2,3)}$.

Two scatter plots are presented. The first plot demonstrates the attacker's perspective on a linkage attack: it visualizes how an attacker attempts to match external and anonymized datasets. Data points are color-coded: blue for the external dataset (K_o, T_o), with T_o omitted as 0 since the attacker do not observe target value for original dataset; red points are for the anonymized dataset (K_a, T_a).

Notice that for the individual with $Id_o = 9$ and a key value of 4.3, no anonymized key falls within the same equivalence class, preventing attackers from making a match (recall that equivalence classes are constructed based on the anonymized data set). As a result, they will not predict the target value for this individual. Consequently, since

Table 2.16: Original Dataset X

Id_O	**key** (K_o)	**target** (T_o)
1	0.5	0.5
2	1.3	0.5
3	1.5	2.3
4	2.1	1.3
5	2.2	1.4
6	2.5	2.6
7	3.1	1.7
8	3.3	1.8
9	4.3	2.2

Table 2.17: External Dataset Z

Id_O	**key** (K_o)
1	0.5
2	1.3
3	1.5
4	2.1
5	2.2
6	2.5
7	3.1
8	3.3
9	4.3

Table 2.18: Anonymized Dataset Y

Id_A	**key** (K_a)	**target** (T_a)
1	0.3	0.7
2	0.7	2.5
3	1.5	1.5
4	2.3	1.7
5	2.8	2.5
6	3.3	1.7
7	3.7	2.7

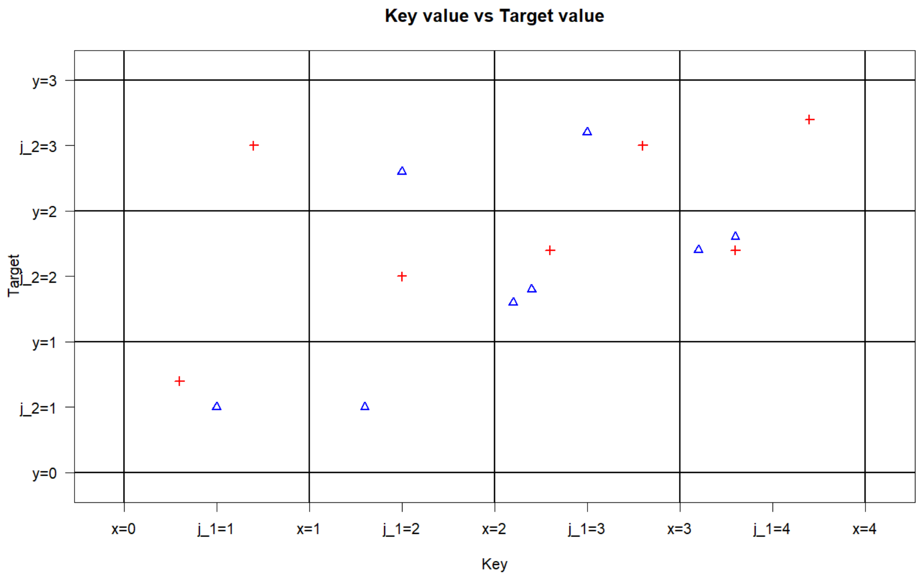


Figure 2.2: Calculation of CAP

the probability of correct prediction cannot be calculated for $Id_o = 9$, they must be excluded from the external data set during this calculation. The second graph illustrates the calculation of the probability of a correct prediction when comparing the original and anonymized data sets. The data points are color-coded: blue for the original dataset (K_o, T_o) and red for the anonymized dataset (K_a, T_a). When blue points and red points share the same grid, it is counted as a correct match. When blue points and red points share the same key value bin, it is counted as a match.

We illustrate the attacker's prediction strategy for the following selected equivalence classes:

- **Equivalence Class $K_{[0,1)}$, Target $T_{O,1}$ with Individual 1:** For Individual 1, with $K_{O,1} = 0.5$ and $T_{O,1} = 0.5$, the attacker observes $K_{O,1} = 0.5$, which belongs to the $K_{[0,1)}$ equivalence class. Upon searching the anonymized dataset, the attacker finds $K_{A,1} = 0.3$ within the same key equivalence class. The predicted value for $T_{O,1}$ is $T_{A,1} = 0.7$, which also falls within the $T_{[0,1)}$ category. This category matches the original target's equivalence class $T_{O,1} = 0.5$, resulting in the correct prediction.

- **Equivalence Class $K_{[1,2)}$, Target $T_{O,2}$ with Individual 2:** For Individual 2, with $K_{O,2} = 1.3$ and $T_{O,2} = 0.5$, the attacker observes $K_{O,2} = 1.3$, placing it in the $K_{[1,2)}$ equivalence class. In the anonymized dataset, the attacker matches this with $K_{A,3} = 1.5$, which also falls within the $K_{[1,2)}$ class. However, the predicted target $T_{A,3} = 1.5$ falls into the $T_{[1,2)}$ category, different from the original target's class $T_{[0,1)}$, leading to an incorrect prediction.

- **Equivalence Class $K_{[3,4)}$, Target $T_{O,7}, T_{O,8}$ with Individuals 7 and 8:** For Individual 7, with $K_{O,7} = 3.1$ and $T_{O,7} = 1.7$, and for Individual 8, with $K_{O,8} = 3.3$ and $T_{O,8} = 1.8$, the attacker observes $K_{O,7} = 3.1$ and $K_{O,8} = 3.3$, both within the $K_{[3,4)}$ equivalence class. In the anonymized dataset, the attacker finds $K_{A,6} = 3.3$ and $K_{A,7} = 3.7$, which also fall within the $K_{[3,4)}$ class. The predicted targets are $T_{A,6} = 1.7$ or $T_{A,7} = 2.7$, corresponding to the $T_{[1,2)}$ and $T_{[2,3)}$ classes, respectively, with an equal probability of $1/2$ for each. One of these predictions matches the original target class $T_{[2,3)}$, resulting in a $1/2$ probability of a correct prediction.

$\square$

In what follows, we provide computations that justify the interpretation of (2.8) as the probability of the correct prediction. We consider the equivalence class $[j_1]$ such that $\exists\ K_{O,i} \in K_{[j_1]}$ and $K_{A,i'} \in K_{[j_1]}$. This represents the scenario where an attacker can match within the key equivalence class $[j_1]$. We denote by $T_{[j']}$ the value predicted by the attacker (the attacker does not predict the specific value, rather than the equivalence class). The probability of correctly predicting the target value for individuals whose key value falls within the equivalence class $K_{[j_1]}$ and whose target value falls within $T_{[j_2]}$ is given by:

$$1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\}\mathbb{P}(T_{[j']} \in T_{[j_2]} \mid K_a \in K_{[j_1]}). \tag{2.9}$$

First, we need to verify that a target-key pair from the original dataset belongs to the $[j_1], [j_2]$ equivalence class. If no such individual exists in the original dataset, the probability of a correct prediction will be zero. The next term $\mathbb{P}(T_{[j']} \in T_{[j_2]} \mid K_a \in K_{[j_1]})$ represents the probability that the predicted target value $T_{[j']}$ falls within $T_{[j_2]}$, which corresponds to correctly predicting the target value. The condition $K_a \in K_{[j_1]}$ reflects the situation in which the attacker matches. Then, the expression in (2.9) equals to

$$\begin{aligned}
&1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\} \sum_{[j_2]'} \mathbb{P}(\{T_{[j']} \in T_{[j_2]}\} \cap \{T_a \in T_{[j_2]'}\} \mid K_a \in K_{[j_1]}) \\
&= 1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\} \\
&\quad \times \sum_{[j_2]'} \mathbb{P}(T_{j'} \in T_{[j_2]} \mid \{T_a \in T_{[j_2]'}\} \cap \{K_a \in K_{[j_1]}\})\mathbb{P}(T_a \in T_{[j_2]'} \mid K_a \in K_{[j_1]}).
\end{aligned}$$

The above equality holds because each bin $T_{[j_2]'}$ is mutually exclusive and the union of all $T_{[j_2]'}$ bins covers the entire set T_a. We note that when the attacker observes $T_a \in T_{[j_2]'}$ and $K_a \in K_{[j_1]}$ for a fixed $[j_2']$ and a fixed $[j_1]$, it indicates that the pair of random variables K_a and T_a fall into specific equivalence classes. Given this observation, the attacker will predict $T_{[j']} = T_{[j_2]'}$ with probability one. This means that $T_{j'} \in T_{[j_2]}$ if and only if $T_a \in T_{[j_2]}$. Thus,

$$\mathbb{P}(T_{j'} \in T_{[j_2]} \mid T_a \in T_{[j_2]'} \cap K_a \in K_{[j_1]}) = \begin{cases} 0 & \text{if } T_{[j_2]'} \neq T_{[j_2]}, \\ 1 & \text{if } T_{[j_2]'} = T_{[j_2]}. \end{cases}$$

Finally, we have

$$\begin{aligned}&1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\}\mathbb{P}(T_{[j']} \in T_{[j_2]} \mid K_a \in K_{[j_1]})\\&= 1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\}\mathbb{P}(T_a \in T_{[j_2]} \mid K_a \in K_{[j_1]}).\end{aligned}$$

We notice that the probability

$$\mathbb{P}(T_o \in T_{[j']} \mid \{T_a \in T_{[j_2]}\} \cap \{K_a \in K_{[j_1]}\})$$

could vary under different problem configurations. It depends on the level of information that an attacker obtains; we will demonstrate this in the next Example 2.6.6.

Example 2.6.6. We revisit Example 2.6.1 with some modifications to demonstrate the calculation of the proposed individual CAP score and how the different levels of information available to an attacker influence the CAP score. We have $\{1, 1\}$ corresponding to the equivalence class (M, P); $\{1, 2\}$ corresponding to the equivalence class (M, N); $\{2, 1\}$ corresponding to the equivalence class (F, N); and $\{2, 2\}$ corresponding to the equivalence class (F, P). In the anonymized dataset Y, notice that the attacker does not

Table 2.19: Original Dataset

Id_O	Original key (Gender)	Original target (Test Result)
1	M	P
2	M	P
3	F	N
4	F	N

Table 2.20: Anonymized Dataset

Id_A	Anonymized key (Gender)	Anonymized target (Test Result)
1	M	N
2	F	N
3	F	N
4	F	N

have access to Id_O, the true identifiers of the original dataset. Identifiers Id_A are shown solely for demonstration purposes and do not correspond to the original identifiers.

- $\text{CAP}_{O,[j_1],[j_2]}$
  - $[j_1] = 1, [j_2] = 1$: $\text{CAP}_{O,[j_1]=1,[j_2]=1} = \frac{2}{2} = 1$,
  - $[j_1] = 1, [j_2] = 2$: $\text{CAP}_{O,[j_1]=1,[j_2]=2} = \frac{0}{2} = 0$,
  - $[j_1] = 2, [j_2] = 1$: $\text{CAP}_{O,[j_1]=2,[j_2]=2} = \frac{0}{2} = 0$,
  - $[j_1] = 2, [j_2] = 2$: $\text{CAP}_{O,[j_1]=2,[j_2]=2} = \frac{2}{2} = 1$.

- **Anonymized Dataset Y:**
  - $[j_1]=1, [j_2]=1$: $\text{CAP}_{A,[j_1]=1,[j_2]=1} = \frac{0}{1} = 1$,
  - $[j_1]=1, [j_2]=2$: $\text{CAP}_{A,[j_1]=1,[j_2]=2} = \frac{0}{1} = 0$,
  - $[j_1]=2, [j_2]=1$: $\text{CAP}_{A,[j_1]=2,[j_2]=1} = \frac{0}{3} = 0$,
  - $[j_1]=2, [j_2]=2$: $\text{CAP}_{A,[j_1]=2,[j_2]=2} = \frac{3}{3} = 1$.

1. **High Knowledge Scenario:** If the attacker has more knowledge of the dataset, such as knowing that all men have a positive result, then the probability that any prediction $T_{[j']}$ belongs to positive, given the observing male, is 1 regardless of the anonymized target equivalence class $T_a \in T_{[j_2]'}$, thus $\mathbb{P}(T_{j'} = \mathbb{P} \mid T_a \in M) = 1$. This is represented by the following summation:

$$\begin{aligned}
&1\{(K_o, T_o) \in (K_{[j_1]}, T_{[j_2]})\}\mathbb{P}(T_{[j']} \in T_{[j_2]} \mid K_a \in K_{[j_1]}) \\
&= \sum_{[j_2]'} \mathbb{P}(T_{j'} \in T_{[j_2]} \mid \{T_a \in T_{[j_2]'}\} \cap \{K_a \in K_{[j_1]}\})\mathbb{P}(T_a \in T_{[j_2]'} \mid K_a \in K_{[j_1]}) \\
&= \mathbb{P}(T_{j'} = P \mid T_a \in M)\mathbb{P}(T_a \in P \mid K_a \in M) \\
&\quad + \mathbb{P}(T_{j'} = N \mid T_a \in M)\mathbb{P}(T_a \in N \mid K_a \in M) \\
&= 1 + 0 = 1.
\end{aligned}$$

2. **Low Knowledge Scenario:** If the attacker has limited information, such as knowledge of a larger external dataset with some names associated with the female equivalence class, Assume that there is a 50% chance that these individuals are included in the anonymized data set. If they are included, the predictions are based on the previous CAP set-up, as previously calculated: $\text{CAP}_{A,[j_1]=2,[j_2]=2} = \frac{3}{3} = 1$. If they are not included, the attacker assumes equal probabilities for positive and negative outcomes, resulting in the correct prediction probability of $\frac{1}{2}$. Thus, the overall probability is calculated as:

Table 2.21: Z

Id_O	Original key (Gender)
3	F
4	F
5	F
6	F

$$\mathbb{P}(T_{[j']} = N \mid T_a \in N \cap K_o \in F) = \frac{1}{2} \times 1 + \left(\frac{1}{2} \times \frac{1}{2}\right) = \frac{3}{4}.$$

Since there is no T_a that is P, the overall probability of making a correct prediction in this scenario is:

$$1\{T_o \in N\} \sum_{j_2} \mathbb{P}(T_{j'} \in N \mid T_a \in T_{j_2} \cap \ K_a \in F)\mathbb{P}(T_a \in j_2 \mid \ K_a \in F)$$
$$= 1\{T_o \in N\}\mathbb{P}(T_{j'} \in N \mid T_a \in N \cap \ K_a \in F)\mathbb{P}(T_a \in N \mid \ K_a \in F) = 3/4.$$

This has a lower probability of correct prediction compared to the CAP setup: $\text{CAP}_{A,[j_1]=2,[j_2]=1} = \frac{3}{3} = 1$.

□

In conclusion, the probability of the correct predictions of the attacker are dependent on the level of information available to the attacker. When the attacker has more knowledge about the results associated with each gender in the dataset, the predictions are more accurate, resulting in a higher CAP score. In contrast, when the attacker lacks detailed information about the presence of specific individuals or their test results, making the correct predictions becomes more complex and uncertain, leading to greater variability in the CAP score.

2.6.5 Population-level equivalence classes CAP

Under a linkage attack, when an anonymized dataset is released, the attackers have information about a pair of variables T_a, K_a and an external dataset that contains a pair of variables K_o and explicit identifiers (EI). In the CAP framework, it is assumed that the keys K_o and the explicit identifier pair (EI) in the external dataset are exactly the same as the pairs in the original dataset.
For each individual i identified within a specific key equivalence class $K_{[j_1]}$ in the external dataset, the attacker attempts to find a match in the anonymized dataset where $K_{A,k} \in K_{[j_1]}$ for one or more k. Upon successful identification of such matches, the predicted value of $T_{O,i}$ is assigned based on one of the corresponding values $T_{A,k}$ from the matched entries. The selection of $T_{A,k}$ depends on the distribution of pairs $(T_{A,k}, K_{A,k})$.

Recall that the attacker sees the external dataset, not the original dataset. Therefore, the disclosure risk measure should be based on a subset of the former one (called the matching set). We denote the number of data in the matching dataset as $n_{O,m}$. We only consider equivalence classes j_1 such that there exist i, i' such that $K_{O,i} \in K_{[j_1]}$ and $K_{A,i'} \in K_{[j_1]}$, as this is the situation in which an attacker will match the equivalence class $[j_1]$. In order to calculate the risk of disclosure at the dataset level, we need to compare the anonymized dataset to the matching dataset. We have the probability of a correct prediction:

$$\mathbb{P}_a(\text{correct match}) \tag{2.10}$$
$$= \sum_{[j_1]=1}^{J_1} \sum_{[j_2]=1}^{J_2} \left(\frac{\sum_{i=1}^{n_{O,m}} 1\{K_{O,i} \in K_{[j_1]}, T_{O,i} \in T_{[j_2]}\}}{n_{O,m}} \right) \cdot \left(\frac{\sum_{k=1}^{n_A} 1\{T_{A,k} \in T_{[j_2]}, K_{A,k} \in K_{[j_1]}\}}{\sum_{k=1}^{n_A} 1\{K_{A,k} \in K_{[j_1]}\}} \right).$$

This formula is an approximation of:

$$\sum_{[j_1]} \sum_{[j_2]} \mathbb{P}(\{K_o \in K_{[j_1]}\} \cap \{T_o \in T_{[j_2]}\}) \mathbb{P}(T_a \in T_{[j_2]} \mid K_a \in K_{[j_1]}).$$

This formula sums up all possible equivalence classes for keys and targets, calculating the product of two probabilities for each combination. The first probability, $\mathbb{P}(K_o \in K_{[j_1]} \cap T_o \in T_{[j_2]})$, represents the proportion of individuals in the matching dataset that fall into the equivalence class of the target key $K_{[j_1]}, T_{[j_2]}$. The second probability, $\mathbb{P}(T_a \in T_{[j_2]} \mid K_a \in K_{[j_1]})$, is the conditional probability that, given that an individual in the anonymized data set belongs to the key class $K_{[j_1]}$, their target also matches the class $T_{[j_2]}$. Since equivalence classes are mutually exclusive, summing all possible equivalence classes of these probabilities provides the overall probability that a target value in the original data can be correctly identified.

We can also calculate the probability of correct prediction when the original dataset is released as a baseline. When the original data set is released, the attacker has information about a pair of variables T_o, K_o and an external dataset that contains K_O and explicit identifiers EI. The probability of a correct prediction in this scenario is:

$$\mathbb{P}_o(\text{correct match}) \tag{2.11}$$
$$= \sum_{[j_1]=1}^{J_1} \sum_{[j_2]=1}^{J_2} \left(\frac{\sum_{i=1}^{n_O} 1\{K_{O,i} \in K_{[j_1]}, T_{O,i} \in T_{[j_2]}\}}{n_O} \right) \cdot \left(\frac{\sum_{k=1}^{n_O} 1\{T_{O,k} \in T_{[j_2]}, K_{O,k} \in K_{[j_2]}\}}{\sum_{k=1}^{n_O} 1\{K_{O,k} \in K_{[j_2]}\}} \right).$$

Note that, since this serves as the baseline for correct predictions in the anonymized dataset, the equivalence classes $[j_1]$ and $[j_2]$ are the same as those in the anonymized dataset.

Example 2.6.7. For Example 2.6.1, we demonstrate the calculation for the score (2.10).

- The original database X contains $n_O = 4$ records.
- j correspond to person in original data with $Id_{O,j}$.
- i correspond to person in anonymized data with $Id_{A,i}$.
- $K_{O,j}$ can take the value (M, F).
- $T_{O,j}$ can take the value (P, N).

Table 2.22: X

Id_O	Original key, Gender	Original target, Test Result
1	M	P
2	M	P
3	F	P
4	F	N

Table 2.23: Z

Id_O	Original key, Gender
1	M
2	M
3	F
4	F

The data holder can see the original data X, Table 2.22, and Y is the anonymized data set. An adversary can see table Z, Table 2.23, and Y Table 2.24. Note that Id_A does not correspond to any explicit identifier (EI) in the original dataset; it is used here solely for demonstration purposes. Note that all individuals in the external data set can be matched, so the matching data set is identical to the external data set.

- Equivalence classes for the key K: $K_{[M]}$ for $M, K_{[F]}$ for F.
- Equivalence classes for the target T: $T_{[P]}$ for $P, T_{[N]}$ for N.
- Calculate the first term $\mathbb{P}(K_o \in K_{[j_1]} \cap T_o \in T_{[j_2]})$:

$$\mathbb{P}(K_o \in K_{[M]} \cap T_o \in T_{[P]}) = \frac{2}{4} = 0.5,$$
$$\mathbb{P}(K_o \in K_{[F]} \cap T_o \in T_{[P]}) = \frac{1}{4} = 0.25,$$
$$\mathbb{P}(K_o \in K_{[F]} \cap T_o \in T_{[N]}) = \frac{1}{4} = 0.25.$$

Table 2.24: Y

Id_A	Anonymized key, Gender	Anonymized target, Test Result
1	M	P
2	F	P
3	F	N
4	F	N

- Calculate the second term $\mathbb{P}(T_a \in T_{[j_2]} \mid K_a \in K_{[j_1]})$:

$$\begin{aligned}\mathbb{P}(T_a \in T_{[P]} \mid K_a \in K_{[M]}) &= \frac{1}{1} = 1,\\ \mathbb{P}(T_a \in T_{[P]} \mid K_a \in K_{[F]}) &= \frac{1}{3} \approx 0.33,\\ \mathbb{P}(T_a \in T_{[N]} \mid K_a \in K_{[F]}) &= \frac{2}{3} \approx 0.67.\end{aligned}$$

Using the formula (2.10):

$$\mathbb{P}_a(\text{correct match}) = (0.5 \times 1) + (0.25 \times 0.33) + (0.25 \times 0.67) = 0.75.$$

The overall probability of correct prediction for the dataset disclosure risk in this example is 0.75. □

Comparison Between Original and Anonymized Disclosure Risk

The score (2.10) provides a proper interpretation as a probability, allowing us to assess the likelihood of correct prediction. However, to effectively evaluate the level of privacy that the anonymized dataset offers, it is crucial to establish a baseline for comparison. Here, we use the probability of correct prediction for the original data as the baseline.

Change in Probability of Attribute Disclosure. To assess the differences in the CAP dataset scores between the anonymized and original datasets, we use equation (2.10) and equation (2.11):

$$\text{CAP}_{\text{change}} = \mathbb{P}_a(\text{correct match}) - \mathbb{P}(\text{correct match})_o. \tag{2.12}$$

The $\text{CAP}_{\text{change}}$ quantifies the overall probability change of making a correct prediction. This measure helps to determine the effectiveness of the anonymization process in reducing the risk of attribute disclosure. However, in some cases, the change for particular equivalence classes could be zero, indicating that there is no improvement in privacy protection for particular equivalence classes. This scenario may represent a higher privacy risk for a certain group of people.

Min Differences. To address this issue and provide a more alternative assessment, we introduce the measure to calculate the minimum change in the correct prediction probabilities between the anonymized and original datasets across all equivalence classes. Recall the definitions (2.6) and (2.8). Define

$$\text{CAP}_{\text{min}} = \min_{[j_1],[j_2]} \left(\text{CAP}_{A,[j_1],[j_2]} - \text{CAP}_{O,[j_1],[j_2]}\right). \tag{2.13}$$

The measure above considers the worst-case scenario with respect to the risk of individual-level disclosure. It highlights the smallest improvement in privacy provided by the anonymization process in all defined equivalence classes. A value closer to zero or negative in this metric suggests minimal to no privacy improvement for a certain combination of key target equivalence class, indicating potential vulnerabilities in privacy for the anonymization technique used.

2.6.6 Performance of equivalence classes CAP scores

In what follows, we illustrate the performance of our equivalence classes CAP scores. We investigate how does noise addition (Which in principle should increase data privacy) affect CAP scores. In principle, the more noise, the more privacy, the lower CAP score. However, as we will see this picture is not obvious. In fact, the resulting CAP scores are sensitive to the choice of equivalence classes and binning.

Example 2.6.8. The experiment investigates the impact of an additive noise mechanism on Correct Attribution Probability (CAP) scores, aiming to assess how these scores change with increasing levels of privacy. In the experiment, an original dataset is generated in which the key and target variables are sampled from a multivariate normal distribution defined by a mean vector of $\mu = [10, 20]$ and a covariance matrix

$$\Sigma = \begin{bmatrix} 20 & 10 \\ 10 & 20 \end{bmatrix},$$

producing a total of $n = 1000$ observations. To simulate data anonymization and evaluate its impact on privacy, a noise additive method is applied, and a noise with normal distribution is added to the target variable of the original dataset. The noise level is with the mean from 0 to 50 and a standard deviation consistently set at 10, thus creating multiple anonymized datasets, each corresponding to a different level of noise added. The idea being that the more noise added, the better privacy. Hence, the bigger mean of the noise, the better privacy.

For each dataset, equivalence classes are established by cutting the range of key and target variables into bins of equal length, and then CAP scores are calculated to assess the probability of correct identification within these classes. In Figure 2.3, the CAP score for population-level equivalence classes in the original dataset (that is, $\mathbb{P}_o$(correct match) in (2.11)) initially remains stable when noise is added. This is because the original data remain unchanged and any small variations are from the selection of different equivalence classes. Note that the selection of target equivalence class depends only on the target value from the anonymized data set. Therefore, as more noise is introduced, the score will eventually approach zero. This happens because as the noise increases, the anonymized target values diverge more from the original data. Eventually, no original targets match the corresponding target equivalence classes. Consequently, there will be no correct match and the original Figure 2.3 score will drop to zero.

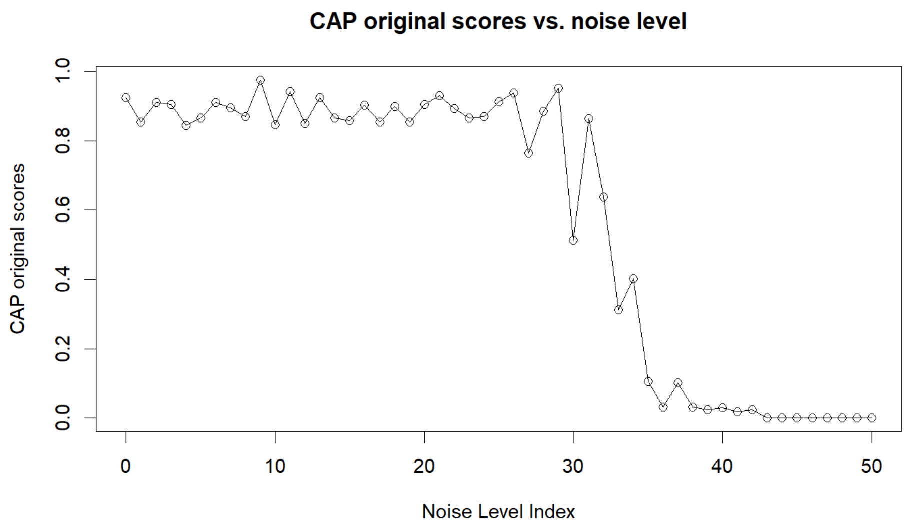


Figure 2.3: Population-level equivalence classes CAP score for the original dataset

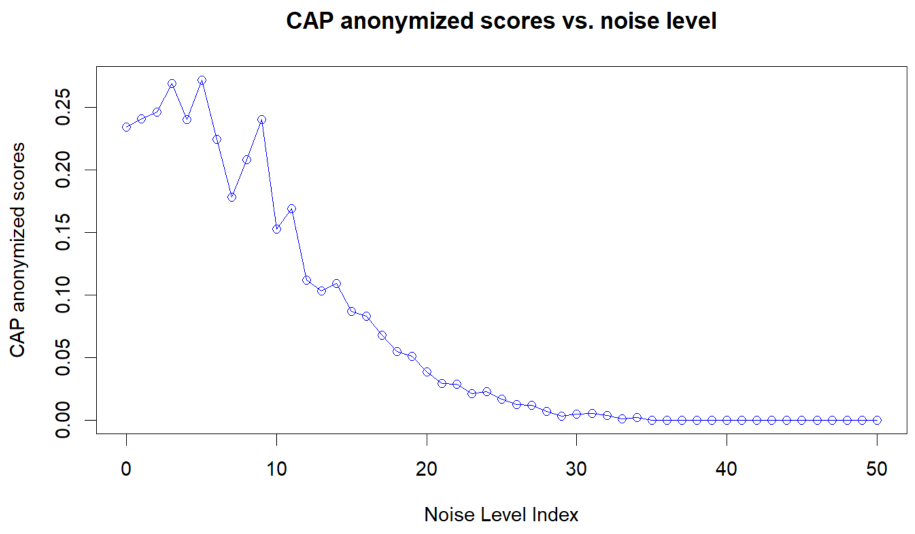


Figure 2.4: Population-level equivalence classes CAP score for the anonymized dataset

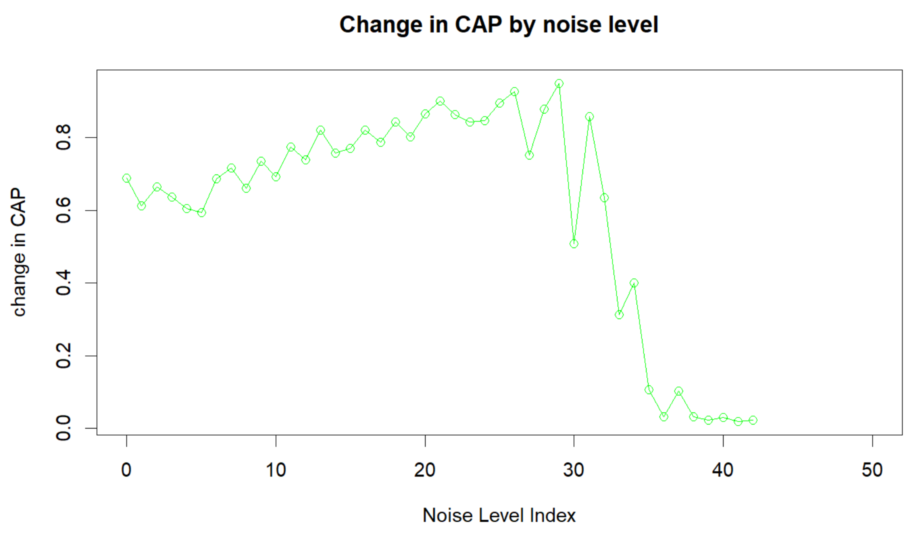


Figure 2.5: CAP score difference between original and anonymized dataset

In Figure 2.4, as more noise is added to the target value, the score $\mathbb{P}_a$(correct match) in (2.10) for the anonymized dataset decreases accordingly, indicating an improvement in privacy.

In Figure 2.5, as more noise is added to the target value, the score $\text{CAP}_{\text{change}}$ in (2.12) increases. However, at a certain point, this difference will drop significantly. This significant drop is due to the decrease in the CAP score of the original dataset that was mentioned earlier.

In Figure 2.6, the CAP_{min} score (Equation (2.13)) shows some values below zero, indicating that adding noise may increase the risk of disclosure and reduce privacy for certain equivalence classes. This is a very strange effect, however, this happens because when only a small amount of noise is introduced, attackers may have a higher probability of making correct predictions for a particular equivalence class, despite a general decrease in disclosure risk across the dataset. Furthermore, CAP_{min} (Equation (2.13)) does not increase significantly as more noise is added. The noticeable jump, observed after adding a substantial amount of noise, is due to the earlier decline in the CAP score of the original dataset. □

2.6.7 Impact of intervals on continuous data

Example 2.6.9. In this example, we illustrate the impact of selecting different intervals for equivalence classes. To demonstrate this effect, we create a dataset containing only continuous data.

The database X contains $n = 4$ entries, as shown in Table 2.25. To create a database Y (Table 2.27), we first consider all entries in X and then apply 2-anonymization with

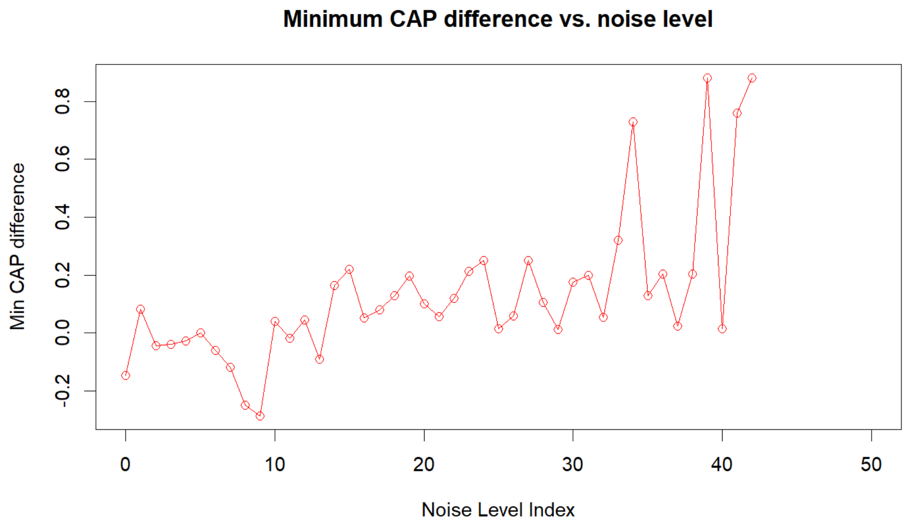


Figure 2.6: Minimum change in the CAP score for each equivalence class between the original and anonymized datasets

Table 2.25: X

Id_O	Original key (Height)	Original target (Weight)
1	160	48
2	161	52
3	168	52
4	170	59

Table 2.26: X

Id_O	Original key (Height)
1	160
2	161
3	168
4	170

Table 2.27: Y

Id_A	Anonymized key (Height)	Anonymized target (Result)
1	160-165	48
2	160-165	52
3	165-170	52
4	165-170	59

respect to height, resulting in 4 records. The target variable remains unchanged in this process.

We select equivalence classes for the key variable according to k-anonymity as follows:

- $K_{[j_1]=1} = (160 - 165)$
- $K_{[j_1]=2} = (165 - 170)$

Next, we choose equivalence classes for the target variable with following interval:

- $T_{[j_2]=1} = 48$
- $T_{[j_2]=2} = 52$
- $T_{[j_2]=3} = 59$
- We now calculate $\text{CAP}_{O,[j_1],[j_2]}$ using formula (2.6), which represents the equivalence class score for the original data. Note that for the $\text{CAP}_{O,[j_1],[j_2]}$ individual score, we only need the original dataset.
- $[j_1] = 1, [j_2] = 1$:
  - Denominator: $K_{[j_1]=1} = (160 - 165)$, with two matching entries in the original dataset.
  - Numerator: $(K_{[j_1]=1}, T_{[j_2]=1}) = (160 - 165, 52)$, with one matching entry.
  - Result: $\text{CAP}_{O,[j_1]=1,[j_2]=1} = 1/2$.
- Similarly, we have $\text{CAP}_{O,[j_1]=1,[j_2]=2} = 1/2$, $\text{CAP}_{O,[j_1]=1,[j_2]=3} = 0$, $\text{CAP}_{O,[j_1]=2,[j_2]=1} = 0$, $\text{CAP}_{O,[j_1]=2,[j_2]=2} = 1/2$ and $\text{CAP}_{O,[j_1]=2,[j_2]=3} = 1/2$.
- We now calculate $\text{CAP}_{A,[j_1],[j_2]}$ using formula (2.8), which represents the equivalence class score for anonymized data.
- $[j_1] = 1, [j_2] = 1$:
  - Denominator: $K_{[j_1]=1} = (160-165)$, with two matching entries in the anonymized dataset.
  - Numerator: $(K_{[j_1]=1}, T_{[j_2]=1}) = (160 - 165, 52)$, with one matching entry.
  - Result: $\text{CAP}_{A,j=1} = 1/2$.
- Similarly, we have $\text{CAP}_{A,[j_1]=1,[j_2]=2} = 1/2$, $\text{CAP}_{A,[j_1]=1,[j_2]=3} = 0$, $\text{CAP}_{A,[j_1]=2,[j_2]=1} = 0$,$\text{CAP}_{A,[j_1]=2,[j_2]=2} = 1/2$ and $\text{CAP}_{A,[j_1]=2,[j_2]=3} = 1/2$.

We can also relax the matching and correct matching criteria so that we can consider equivalence classes for the key variable as $K_{[j_1]=1} = (160 - 170)$ and select equivalence classes for the target variable as $T_{[j_2]=1} = (48 - 59)$.

- We now calculate $\text{CAP}_{O,[j_1],[j_2]}$ using formula (2.6), which represents the equivalence class score for the original data. Note that for the $\text{CAP}_{O,[j_1],[j_2]}$ individual score, we only need the original dataset.

- $[j_1] = 1, [j_2] = 1$:

  - Denominator: $K_{[j_1]=1} = (160-170)$, with four matching entries in the original dataset.
  - Numerator: $(K_{[j_1]=1}, T_{[j_2]=1}) = (160-170, 48-59)$, with four matching entries.
  - Result: $\text{CAP}_{O,[j_1]=1,[j_2]=1} = 1$.

- We now calculate $\text{CAP}_{A,[j_1],[j_2]}$ using formula (2.8), which represents the equivalence class score for anonymized data.

  - Denominator: $K_{[j_1]=1} = (160-170)$, with four matching entries in the anonymized dataset.
  - Numerator: $(K_{[j_1]=1}, T_{[j_2]=1}) = (160-170, 48-59)$, with four matching entries.
  - Result: $\text{CAP}_{A,j=1} = 1$.

□

Comments. Relaxing the matching and correct matching criteria at certain points leads to identical scores between the original and anonymized datasets. As the size of the equivalence classes increases further, the CAP scores for both the original and anonymized datasets converge to 1. This suggests that with larger equivalence classes, the CAP score loses its ability to provide meaningful insights into the dataset's privacy. Essentially, broader classes generalize the data to such an extent that the distinctions between anonymized and original data disappear, making the CAP score ineffective for privacy evaluation.

We can also consider varying the support for key and target intervals based on the information that is presumed to be accessible to the attacker. Here, we assume that the attacker has access to both original and anonymized key values but can only view the anonymized target values. This setup, which was previously introduced, allows the CAP score to be interpreted as the probability of a correct prediction, reflecting the realistic constraints of what the attacker can see and use.

The choice of equivalence class for calculating correct prediction is heavily influenced by how attackers perceive differences between data points. For instance, if an attacker considers heights of 160 cm and 170 cm indistinguishable, they are likely to treat these as a single equivalence class. The selection of equivalence classes generally depends on the willingness to lose information, meaning that when choosing equivalence classes, attackers must consider how much detail can be sacrificed. Broader equivalence classes simplify the data further, reducing specificity, while narrower classes contain more detailed information. From the data owner's perspective, since the attacker's strategy in selecting equivalence classes is unknown, multiple strategies for setting these classes can be adopted to measure data privacy effectively. Some methods include:

1. **Equal length intervals**: Dividing continuous data into intervals of equal length. This is a simple straightforward method; however, it could potentially leave some intervals denser than others.

2. **Equal frequency intervals**: Dividing the data into intervals, each containing a similar number of data points. This method aims to balance the distribution of data across intervals, reducing sparse or overly dense areas.

The goal of this experiment is to test how different binning strategies influence the probability of correct prediction.

Example 2.6.10. An original dataset is created by sampling key and target variables from a multivariate normal distribution, characterized by a mean vector of $\mu = [5, 15]$ and a covariance matrix

$$\Sigma = \begin{bmatrix} 15 & 15 \\ 15 & 15 \end{bmatrix},$$

resulting in a total of $n = 10000$ observations. That is, we have a vector of perfectly dependent normals, with different means. To simulate data anonymization and assess its impact on privacy, a fixed exponential noise is added to the target variable of the original data set with a rate parameter of 0.5, generating an anonymized dataset with altered target values and the same key values.

Equivalence classes are then constructed by cutting the ranges of key and target variables into different bin sizes, number of bins from 1 to 10. This procedure is applied with two different binning methods: equal length and equal frequency. CAP scores are calculated for each selection of the binning method to evaluate the probability of correct prediction within different classes. From Figure 2.7, as the binning size increases, the CAP score decreases. This is because larger equivalence classes include a wider range of data, which decrease the precision of the information within each class. As the criteria for making and claiming a correct match become less strict, attackers find it easier to make a match and claim that they have a correct match with less precision. For example, if the actual income of a person is \$50,000, tighter equivalence classes might only allow claims around \$45,000-\$55,000 as a match. However, as the equivalence classes expand, an attacker could claim a correct match even if the prediction is as far off as \$30,000-\$80,000.
The use of the equal frequency method leads to a smoother 3D plot, which improves

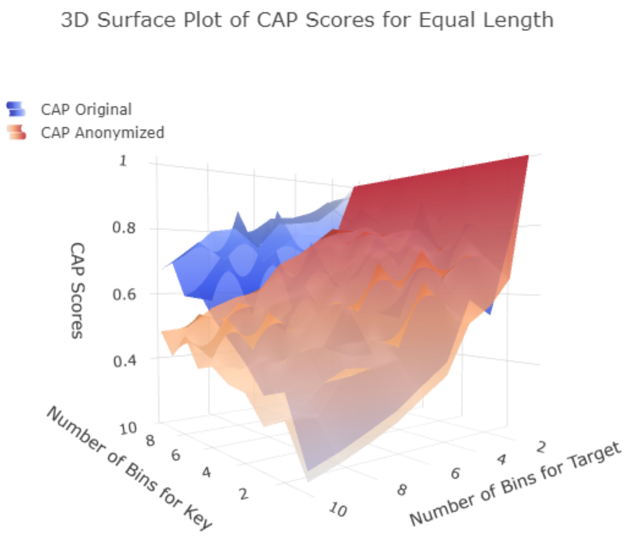


Figure 2.7: 3D plot of equal length method, CAP vs. number of bins for target and key

overall visualization. This smoother representation helps better understand and analyze the changes and patterns in the CAP scores, providing clearer insights into how data anonymization affects the probabilities of attribution.

The following four graphs provide 2D contour plots to visualize Correct Attribution Probability (CAP) scores for original and anonymized datasets under two binning strategies: equal length and equal frequency: Figure 2.9, Figure 2.10, Figure 2.11, Figure 2.12. Each plot uses shades of blue and red to represent the CAP scores for the original and anonymized data, respectively, mapped against the number of bins for key and target variables.

In conclusion, a smaller number of bins group data into broader categories, resulting in a higher CAP score; this does not necessarily imply reduced privacy. The choice of binning strategy affects the CAP score, and the scores from different binning strategies are not directly comparable. □

As demonstrated in the previous example, the CAP scores change depending on the number of bins used for the key and target variables. To effectively compare different anonymization algorithms especially applied to different original datasets, it is important to evaluate CAP scores with varying numbers of bins. By dividing key and target attributes into different bin counts, we can see CAP scores under different numbers of bins.

Example 2.6.11. The experiment is designed to analyze probability of correct prediction across two distinct datasets. The first dataset is generated by sampling 10,000 observations from a multivariate normal distribution with mean vector $\mu = [5, 15]$ and

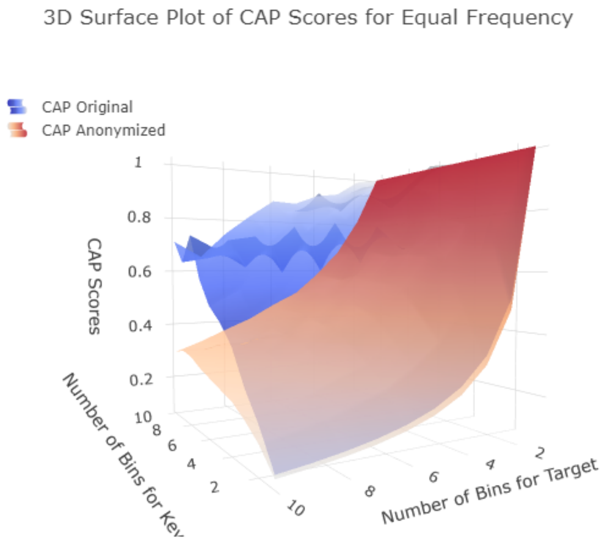


Figure 2.8: 3D plot of equal frequency method, CAP vs. number of bins for target and key

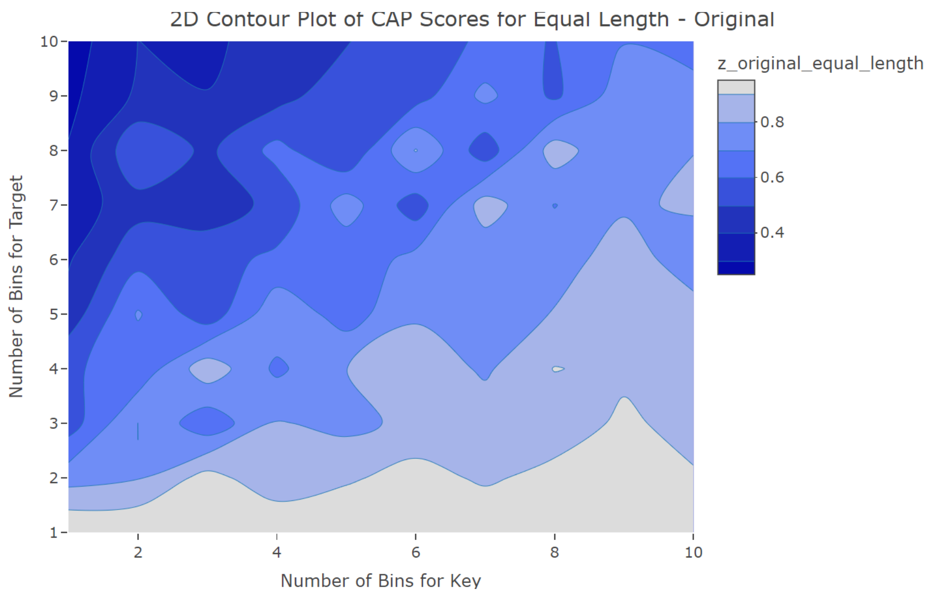


Figure 2.9: 2D plot of equal length method for original data, CAP vs. number of bins for target and key

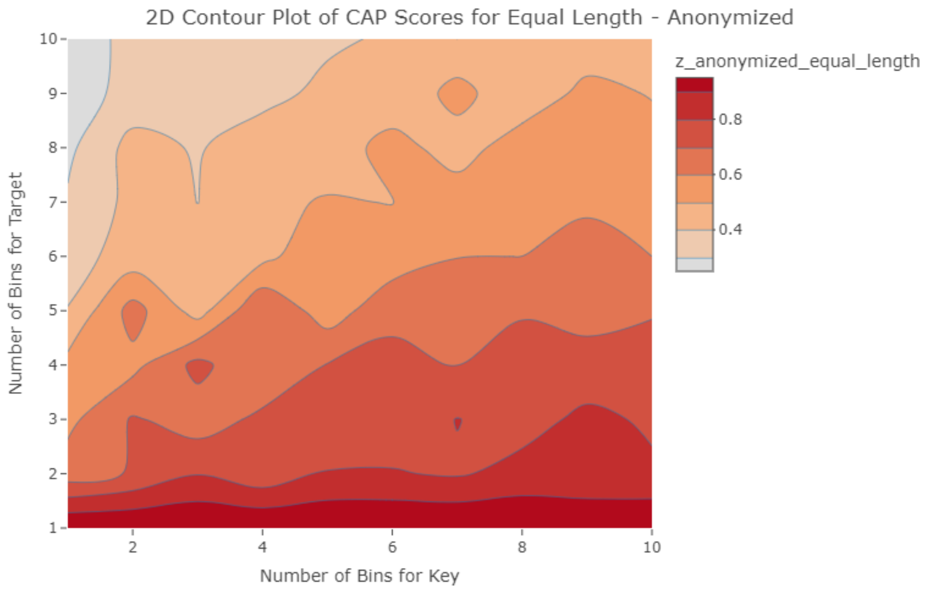


Figure 2.10: 2D plot of equal length method for anonymized data, CAP vs. number of bins for target and key

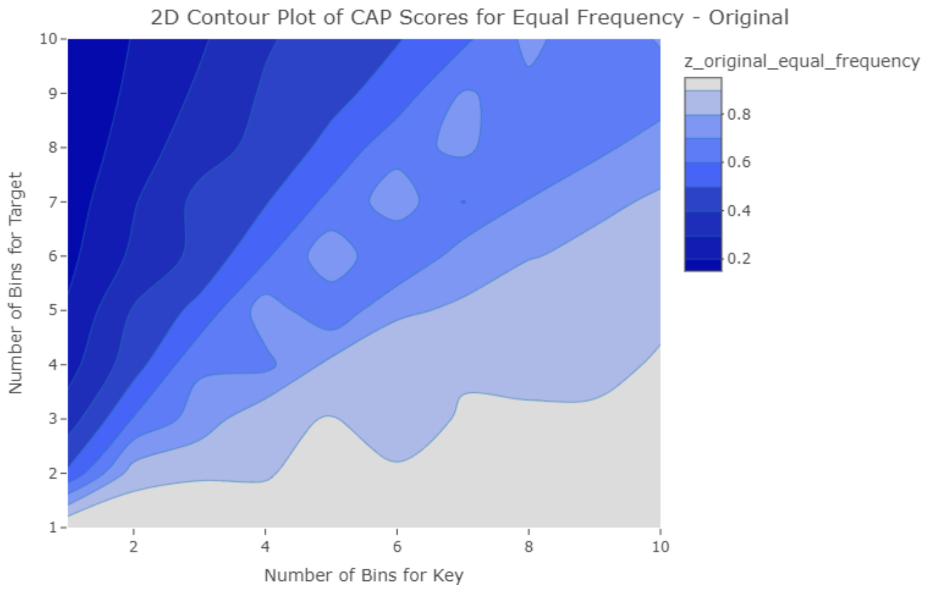


Figure 2.11: 2D plot of equal frequency method for original data, CAP vs. number of bins for target and key

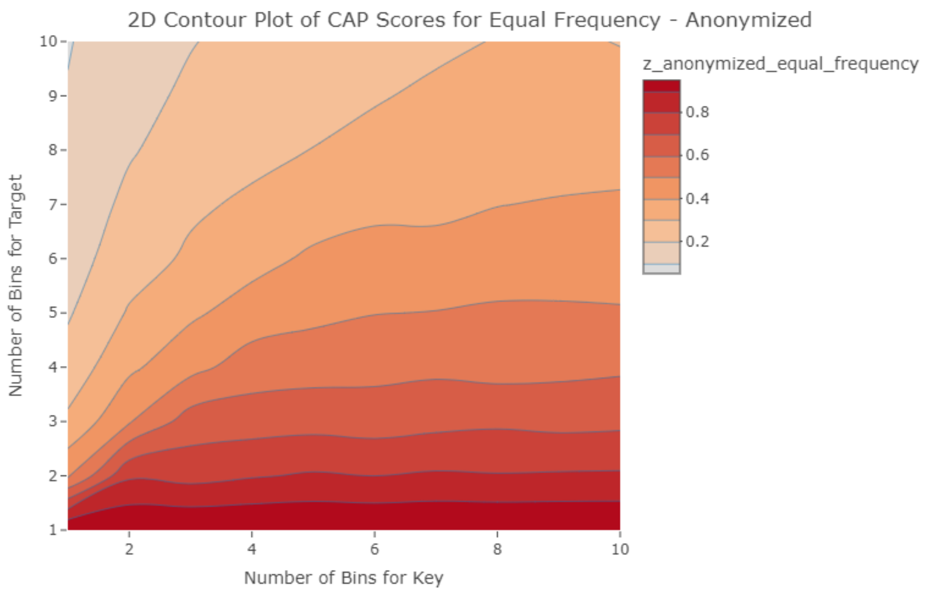


Figure 2.12: 2D plot of equal frequency method for anonymized data, CAP vs. number of bins for target and key

covariance matrix

$$\Sigma = \begin{bmatrix} 15 & 15 \\ 15 & 15 \end{bmatrix}.$$

Similarly, the second dataset is produced with a mean vector $\mu = [1, 5]$ and a simpler covariance matrix

$$\Sigma = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}.$$

In both cases, the target variable is modified by adding noise drawn from an exponential distribution, with the first dataset undergoing modifications with a rate parameter of 0.5 and the second with a rate parameter of 5. This procedure aims to create two levels of anonymized data sets, each reflecting a different addition of noise. In this case, we cannot justify which anonymized dataset is more private. To address this, we employ a 3D plot to visually compare the CAP scores across different bin sizes and noise levels. $\square$

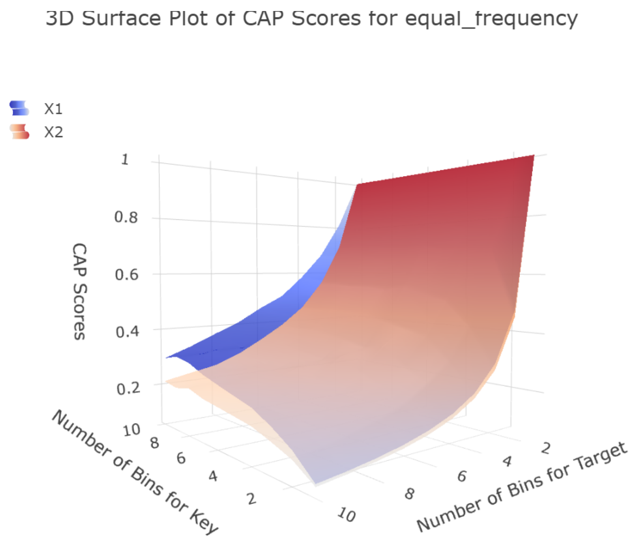


Figure 2.13: Comparison of CAP anonymized score between 2 randomization algorithm

Chapter 3

Synthetic Data Generation

In data generation, the methodology can be determined based on knowledge of the density or cumulative distribution functions (CDFs) of the data set. The process can be described as follows:

- **Exact knowledge of density or CDF**: If exact density or cumulative distribution functions are known, proceed directly to techniques such as inverse transformation or acceptance-rejection methods.
- **Some knowledge of density or CDF, or a model**: If we know that a density or CDF comes from a particular family, but we do not know their parameters, we can proceed as follows. Estimate the parameters and apply the methods mentioned above. Furthermore, we may have knowledge of a parametric model (for example, a regression model). This leads to parametric-based data generation methods.
- **No assumptions or knowledge**: If neither the density nor the CDF is known, a non-parametric approach is appropriate. This involves using methods such as bootstrap, SMOTE, or linear piecewise techniques.

Most of the methods are standard methods that can be found in textbooks. There are some exceptions, for example SMOTE is relatively less known; see [4].

3.1 Terminology

In what follows, we will need the following terminology.

- F is the cumulative distribution function (CDF) of a univariate random variable X .
- f is the density of F (if it exists).

- The left and right endpoints a, b of F are defined by $a = \inf\{t \in R : F(t) > 0\}$, $b = \sup\{t \in R : F(t) < 1\}$.
- The quantile function or generalized inverse function of F is defined as

$$Q(u) := F^{\leftarrow}(u) := \inf\{x \in \mathbb{R} : F(x) \geq u\} .$$

- If F is strictly increasing and continuous then the generalized inverse is equal to the classical inverse. We then denote the quantile function as F^{-1} .
- If we need to emphasize which random variable we are dealing with, we are going to write F_X, Q_X, F_X^{-1}, $F_X^{\leftarrow}$.
- F_n is the empirical distribution based on data $X_1, \ldots, X_n$, defined as

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{[X_i,\infty)}(x).$$

3.2 Exact Knowledge of Density

3.2.1 Inverse Transformation for continuous random variables

Proposition 3.2.1. *If F is continuous and strictly increasing on (a, b), then F^{-1}, the inverse of F, exists and is also continuous and strictly increasing. Furthermore*

- *Let a and b be the left and the right endpoints of F. Then*

$$\lim_{u \to 0^+} F^{-1}(u) = a, \quad \lim_{u \to 1^-} F^{-1}(u) = b.$$

  The results hold also if $a = -\infty$ or $b = +\infty$.
- *For $x, x' \in [a, b]$, $F^{-1}(F(x)) = F^{-1}(F(x')) \Rightarrow x = x'$.*
- *For all $x \in (0, 1)$, we have $F(F^{-1}(x)) = x$.*
- *$F(x) = u \Leftrightarrow x = F^{-1}(u)$, for all $x \in (a, b)$ and $u \in (0, 1)$.*

Theorem 3.2.2. *Let F be a CDF. Assume that F is continuous and strictly increasing.*

- *Let U be a random variable, uniformly distributed on $[0, 1]$. Then the CDF of the random variable $F^{-1}(U)$ is F .*
- *Let X be a random variable with CDF F. Then $F(X)$ is uniformly distributed on $[0, 1]$.*

Proof. • Let U be a random variable, uniformly distributed on $[0, 1]$. Then for $x \in (a, b)$,

$$\mathbb{P}(F^{-1}(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x).$$

- Next, for $x \in [0, 1]$,

$$\begin{aligned}\mathbb{P}(F(X) \leq x) &= \mathbb{P}(F^{-1}(F(X)) \leq F^{-1}(x)) \\ &= \mathbb{P}(X \leq F^{-1}(x)) = F(F^{-1}(x)) = x.\end{aligned}$$

  Thus, $F(X)$ has a uniform distribution.

□

Simulation Algorithm

- Generate a sample of size n from the uniform distribution on $[0, 1]$. Denote it by u_i, $i = 1, \ldots, n$.
- For each u_i, calculate $x_i = F^{-1}(u_i)$.
- Then x_i, $i = 1, \ldots, n$, is a sample from the CDF F.

Algorithm complexity The overall complexity depends on the complexity of the inverse function F^{-1}. For univariate data with a closed-form inverse function, the complexity of generating a single $F^{-1}(u_i)$ does not depend on n, denote it as m_1.

- Generating n uniform random numbers between 0 and 1 using the `runif` function takes n operations.
- Assume generating $F^{-1}(u_i)$ for each sample involves m operations or with big O notation $O(1)$.

Thus, the complexity for a sample of size n is $n \times m$ operations or with big O notation $O(n)$.

Example 3.2.3. We want to generate data from an exponential distribution with parameter $\lambda = 1$, that is $F(x) = 1 - e^{-x}$, $x \geq 0$.

- We first find $F^{-1}(p) = -\frac{1}{\lambda} \ln(1 - p)$.
- We generate n observations u_i from the standard uniform distribution.

- Then we calculate $x_i = -\frac{1}{\lambda} \ln(1 - u_i)$ for each i.

The overall complexity depends on the following steps:

- Generating n uniform random numbers u_i using the **runif** function takes n operations, with a complexity of $O(n)$.
- Calculating $-\frac{1}{\lambda} \ln(1 - u_i)$ for each i involves one logarithm calculation and one multiplication. Assume takes constant number of operation m_1, the complexity of each logarithm calculation is $O(1)$.
- Thus, for a sample of size n, it takes nm_1 operations, resulting in a total complexity of $O(n)$.

Assume the complexity per number generated is m_1 for generating a logarithmic calculation, resulting in a total complexity of $m_1 \times n$ for a sample of size n, or with a big O notation, $O(n)$. $\square$

3.2.2 Inverse transformation for discrete random variables

We want to generate random numbers from a non-continuous random variable X. This means that the associated cumulative distribution function (CDF) F is right-continuous only, but not continuous. Also the CDF is not strictly increasing. In this case, the inverse function F^{-1} is not defined, but the generalized inverse Q is well defined. Here, F is assumed to be known.

Example 3.2.4. Consider a random variable X where $\mathbb{P}(X = 0) = \mathbb{P}(X = 1) = \frac{1}{2}$. The cumulative distribution function (CDF) and the quantile function for X are as follows: The CDF, $F(x)$, is given by:

$$F(x) = \begin{cases} 0 & \text{if } x < 0, \\ \frac{1}{2} & \text{if } 0 \leq x < 1, \\ 1 & \text{if } x \geq 1. \end{cases}$$

The quantile function, $Q_X(p)$, is given by:

$$Q(p) = \begin{cases} 0 & \text{if } 0 \leq p < \frac{1}{2}, \\ 1 & \text{if } \frac{1}{2} \leq p \leq 1. \end{cases}$$

$\square$

Proposition 3.2.5.

- *Let a and b be the left and right endpoints of F. Then $\lim_{u \to 0} Q(u) = a$, $\lim_{u \to 1} Q(u) = b$. The results also hold if $a = -\infty$ or $b = +\infty$.*
- *For $x, x' \in [a, b]$, $F(x) \leq F(x') \Rightarrow Q(F(x)) \leq x'$.*

- *$F(x) \geq u \Leftrightarrow x \geq Q(u)$, for all $x \in (a,b)$ and $u \in (0,1)$.*

Proof.
- We first prove that Q is left-continuous and non-decreasing. Let $u < v$, as $F(Q(v)) \geq v \Rightarrow F(Q(v)) \geq u$, therefore, $Q(u) \leq Q(v)$, thus Q is non-decreasing.
- Now we prove $F(x) \leq u \Leftrightarrow x \leq Q(u)$. For a given x such that $F(x) \leq u$ and $Q(u)$ is the infimum of all x' such that $F(x') \geq u$, which is greater than or equal to x as F is a non-decreasing function. The reverse is true with a similar proof.

□

Theorem 3.2.6. *Let F be a CDF and let Q be its associated quantile function.*

- *Let U be a random variable, uniformly distributed on $[0,1]$. Then the CDF of the random variable $Q(U)$ is F .*
- *Let X be a random variable with CDF F. Then $Q(X)$ is uniformly distributed on $[0,1]$.*

Proof. We use Proposition 3.2.5.

- Let U be a random variable, uniformly distributed on $[0,1]$. Then for $-\infty < x < \infty$, we have:

$$\mathbb{P}(Q(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x)$$

- Next $x \in [0,1]$:

$$\mathbb{P}(F(X) \leq x) = \mathbb{P}(Q(F(X)) \leq Q(x)) = \mathbb{P}(X \leq Q(x)) = F(Q(x)) = x$$

  This shows that $F(X)$ has the uniform distribution.

□

We present the simulation algorithm for a discrete random variable, where $\mathbb{P}(X = x_i) = f_i$ for $i = 1, \ldots, q$, with $\sum_{i=1}^{q} f_i = 1$ and $f_i > 0$. Then, the CDF is

$$F(x_i) = \mathbb{P}(X \leq x_i) = F_i = \sum_{j=1}^{i} f_j,$$

and $F(x) = F(x_i)$ whenever $x_i \leq x < x_{i+1}$. The corresponding quantile function is given by:

$$Q(u) = x_k \quad \text{where } k = \min\{i : F(x_i) \geq u\} .$$

We have the following algorithm when dealing with a discrete cumulative distribution function that has a finite domain of possible values.

Simulation Algorithm for Finite Domain

- Generate a sample of size n from the uniform distribution on $[0, 1]$. Denote it by u_j, $j = 1, \ldots, n$.
- For each $j = 1, \ldots, n$, if $F_{i-1} < u_j \leq F_i$, then set x_i as a random number from the CDF F.

Algorithm Complexity for Discrete Distribution with Finite Domain The overall complexity of the simulation algorithm depends on the efficiency of determining which interval $[F_{i-1}, F_i]$ the uniform sample u_j falls in, where F_i represents the cumulative probability up to the ith value and u_j are generated from a uniform distribution.

- Generate a sample of size n from the uniform distribution on $[0, 1]$. Denote these samples by u_j, where $j = 1, \ldots, n$. Generating n such numbers has a complexity of $O(n)$.
- For each u_j, determining which interval it falls into requires checking against the intervals defined by the F_i values. The complexity of this step depends on the method used for checking:
  - **Linear Search Complexity:** In a linear search, each uniform sample u_j is compared sequentially with cumulative probabilities $F_1, F_2, \ldots, F_q$. The algorithm starts by checking if $u_j \leq F_1$. If not, it moves to the next interval, checking $u_j \leq F_2$, and so on, until it finds the correct interval $[F_{i-1}, F_i]$ where $F_{i-1} < u_j \leq F_i$. Number of comparisons: The worst-case scenario occurs when the sample is close to F_q, requiring q comparisons. However, because the samples are uniformly distributed, on average a sample u_j will be found around the middle of the distribution. This results in the expected number of comparisons being $q/2$. In big O notation, $O(q/2)$ is equivalent to $O(q)$.
  - **Binary Search Complexity:** In a binary search, each u_j is located in the correct interval by repeatedly dividing the set of intervals in half. The search begins by comparing u_j with the middle of cumulative probability $F_{\lceil q/2 \rceil}$. Depending on whether u_j is less than or greater than this value, the search continues in the left or right half of the interval set. This search process continues until the correct interval is found. Number of comparisons: The binary search requires at most $O(\log_2 q)$ comparisons to find the correct interval for a single u_j. This is because the binary search splits the search space in half with each comparison. After one comparison, there are $q/2$ possible intervals

remaining, after two comparisons, $q/4$, and so on, until the correct interval is identified. The number of splits needed to reduce the set interval to 1 is logarithmic in the number of intervals, hence $O(\log_2 q)$.

Thus, the total complexity is $O(nq)$ using linear search and $O(n \log_2 q)$ using binary search.

Example 3.2.7 (Linear search). Suppose we have a discrete random variable X with three possible values, $x_1 = 1$, $x_2 = 2$, and $x_3 = 3$, and corresponding probabilities $\mathbb{P}(X = x_1) = 0.2$, $\mathbb{P}(X = x_2) = 0.5$, and $\mathbb{P}(X = x_3) = 0.3$. The cumulative distribution function (CDF) values are:

$$F_1 = 0.2, \quad F_2 = 0.7, \quad F_3 = 1.0$$

We generate a uniform random sample $u_j = 0.6$. Using a linear search, we check u_j against each F_i in sequence:

- Check if $F_1 = 0.2$: $0.6 > 0.2$ (move to the next interval)
- Check if $F_2 = 0.7$: $0.6 \leq 0.7$ (stop and select x_2 as the random value corresponding to $u_j = 0.6$)

This method requires checking each interval sequentially until the appropriate one is found, which in this case took 2 checks. □

Example 3.2.8 (Linear vs. binary search). Suppose that we have a discrete random variable X with five possible values, $x_1 = 1$, $x_2 = 2$, $x_3 = 3$, $x_4 = 4$, and $x_5 = 5$, and corresponding probabilities $\mathbb{P}(X = x_1) = 0.1$, $\mathbb{P}(X = x_2) = 0.2$, $\mathbb{P}(X = x_3) = 0.3$, $\mathbb{P}(X = x_4) = 0.2$, and $\mathbb{P}(X = x_5) = 0.2$. The cumulative distribution function (CDF) values are as follows:

$$F_1 = 0.1, \quad F_2 = 0.3, \quad F_3 = 0.6, \quad F_4 = 0.8, \quad F_5 = 1.0$$

We generate three uniform random samples: $u_1 = 0.15$, $u_2 = 0.75$, and $u_3 = 0.95$. Using a linear search, we check each u_j against the F_i values in sequence:

- For $u_1 = 0.15$:
  - Check $F_1 = 0.1$: $0.15 > 0.1$ (move to the next interval)
  - Check $F_2 = 0.3$: $0.15 \leq 0.3$ (select x_2)
  - Total comparisons: 2.
- For $u_2 = 0.75$:
  - Check $F_1 = 0.1$: $0.75 > 0.1$ (move to the next interval)

  - Check $F_2 = 0.3$: $0.75 > 0.3$ (move to the next interval)
  - Check $F_3 = 0.6$: $0.75 > 0.6$ (move to the next interval)
  - Check $F_4 = 0.8$: $0.75 \leq 0.8$ (select x_4)
  - Total comparisons: 4.

- For $u_3 = 0.95$:
  - Check $F_1 = 0.1$: $0.95 > 0.1$ (move to the next interval)
  - Check $F_2 = 0.3$: $0.95 > 0.3$ (move to the next interval)
  - Check $F_3 = 0.6$: $0.95 > 0.6$ (move to the next interval)
  - Check $F_4 = 0.8$: $0.95 > 0.8$ (move to the next interval)
  - Check $F_5 = 1.0$: $0.95 \leq 1.0$ (select x_5)
  - Total comparisons: 5.

In this example, the linear search required 2, 4, and 5 comparisons for each sample, respectively, depending on how far down the list the sample was found.

Using the same discrete random variable X and CDF values as above, we again generate the same uniform random samples: $u_1 = 0.15$, $u_2 = 0.75$, and $u_3 = 0.95$. Using binary search, we perform the following steps:

- For $u_1 = 0.15$:
  - Compare with the middle value $F_3 = 0.6$: $0.15 \leq 0.6$ (search the left half)
  - Compare with $F_2 = 0.3$: $0.15 \leq 0.3$ (search the left half)
  - Compare with $F_1 = 0.1$: $0.15 > 0.1$ (select x_2)
  - Total comparisons: 3.

- For $u_2 = 0.75$:
  - Compare with the middle value $F_3 = 0.6$: $0.75 > 0.6$ (search the right half)
  - Compare with $F_4 = 0.8$: $0.75 \leq 0.8$ (select x_4)
  - Total comparisons: 2.

- For $u_3 = 0.95$:
  - Compare with the middle value $F_3 = 0.6$: $0.95 > 0.6$ (search the right half)
  - Compare with $F_4 = 0.8$: $0.95 > 0.8$ (search the right half)
  - Compare with $F_5 = 1.0$: $0.95 \leq 1.0$ (select x_5)
  - Total comparisons: 3.

In this example, the binary search required 3, 2, and 3 comparisons for each sample, respectively. The binary search is generally more efficient, particularly as the number of intervals q increases. □

Discrete random variables with infinite domain. If we have an infinite domain, the previous algorithm for finite domains is not applicable. The main idea is to replace the finite domain q with a random Q, which depends on the data.

Additional challenges arise from the range of the data. For an infinite domain, we can have two scenarios:

- The range is bounded. For example, a discrete random variable concentrated on rational numbers within $[0, 1]$.
- The range is unbounded (on both sides or one side). For example, a Poisson distribution.

In the second scenario, special care must be taken to manage the "minimal" and "maximal" points effectively.

Simulation Algorithm for Infinite Domain

- Generate a sample of size n from the uniform distribution on $[0, 1]$. Denote these samples by u_j, $j = 1, \ldots, n$.
- Estimate the practical minimum value $x_{\min}$ or maximum value $x_{\max}$ based on the distribution's characteristics. Start from a sufficiently small (for $x_{\min}$) or large (for $x_{\max}$) value and incrementally compute $F(x)$ until $F(x)$ becomes greater than or equal to the smallest or largest generated uniform sample $u_{\min} = \min(u_1, u_2, \ldots, u_n)$ or $u_{\max} = \max(u_1, u_2, \ldots, u_n)$, respectively.
- Starting from $x_{\min}$ (or $x_{\max}$), compute the cumulative distribution function $F(x)$ as the sum of $p(k; \theta)$ for all $k \leq x$ (or $k \geq x$ if starting from $x_{\max}$).
- Continue to increment x and sum the probabilities until $F(x) > u_{\max}$ (if starting from $x_{\min}$) or $F(x) > u_{\min}$ (if starting from $x_{\max}$). This ensures that the computed CDF covers the entire range of the generated uniform samples.
- For each $j = 1, \ldots, n$, if $F_{i-1} < u_j \leq F_i$, then set x_i as a random number corresponding to u_j from the CDF F.

Algorithm Complexity for Discrete Distribution with Infinite Domain The overall complexity of the algorithm depends on the number of intervals q and the number of steps m required to find these intervals. Specifically, generating n uniform samples has a complexity of $O(n)$, and determining the necessary intervals involves m steps, where m accounts for the operations needed to compute and sum the probabilities until the CDF sufficiently covers all uniform samples. Once the intervals are identified, searching through them to find the corresponding x_i for each u_j incurs a complexity of $O(nq)$ for linear search or $O(n \log q)$ for binary search. Therefore, the total complexity is $O(m) + O(q^2) + O(nq)$ for the linear search and $O(m) + O(q^2) + O(n \log q)$ for the binary search, the overall efficiency being influenced by both the interval-finding process and the chosen search method.

Example 3.2.9. We aim to generate data from a Poisson distribution with a rate parameter $\lambda = 3$, where the probability mass function is given by

$$\mathbb{P}(X = k) = \frac{3^k e^{-3}}{k!}$$

for $k \geq 0$.

- Generate a sample of size n from the uniform distribution on $[0, 1]$, denoted by u_j for $j = 1, \ldots, n$.
- Since the minimum value for x is known to be 0, start from $x = 0$ and compute the cumulative distribution function $F(x)$ as the sum of $\mathbb{P}(X = k)$ for all $k \leq x$:

$$F(x) = \sum_{k=0}^{x} \frac{3^k e^{-3}}{k!}$$

- Increase x and continue summing the probabilities until $F(x) > u_{\max}$, where $u_{\max}$ is the largest uniform sample generated.
- For each $j = 1, \ldots, n$, if $F_{i-1} < u_j \leq F_i$, set x_i as the random number corresponding to u_j based on the CDF F.

The complexity of the algorithm involves:

- Computing the cumulative probabilities for q intervals, where q depends on the rate parameter λ. This requires $O(q)$ operations.
- Generating n uniform random numbers, which have a complexity of $O(n)$.
- Search through the cumulative probabilities to find the corresponding intervals for each u_j, which takes $O(n \times q)$ operations for linear search or $O(n \log q)$ for binary search.

Thus, the total complexity is $O(n) + O(q) + O(nq)$ for the linear search and $O(n) + O(q) + O(n \log q)$ for the binary search. In general, complexity depends mainly on the number of intervals q, the efficiency of the search method, and the rate parameter λ. $\square$

3.2.3 Multivariate Inverse Transform Sampling

Our goal is to generate numbers from d-variate distribution by applying the inverse transformation method. This approach requires knowledge of the conditional inverse cumulative distribution functions:

$$F^{-1}_{X_1}, F^{-1}_{X_2|X_1}, \ldots, F^{-1}_{X_d|X_1,\ldots,X_{d-1}}.$$

These functions allow us to convert uniformly distributed random numbers into numbers that follow the target multivariate distribution.

Simulation Algorithm: Multivariate Inverse Transform Sampling

- For each dimension k from 1 to m repeat follow:
  - Generate a random number $u_{i,k}$ from a uniform distribution over $[0,1]$.
  - If $k=1$: compute $x_{i,} = F^{-1}_{X_1}(u_{i,1})$.
  - Otherwise if $k = 2,\ldots,d$: compute $x_{i,k} = F^{-1}_{X_k|X_1,\ldots,X_{k-1}}(u_{i,k}, x_{i,1},\ldots,x_{i,k-1})$.
- Repeat above N times.

Complexity Analysis

- This complexity arises because for each sample, each dimension k may require a different amount of computational effort depending on the complexity of its inverse CDF function, denoted as C_k for $k = 1,\ldots,d$.
- The total computational cost is hence a product of the number of samples (N) and the sum of the complexities of computing the inverse CDF for each dimension. Time Complexity: $O(N \cdot (C_1 + C_2 + \ldots + C_d))$.

Example 3.2.10. Suppose that we want to generate samples from a bivariate distribution follows the following form:

$$f_{X_1}(x_1) = \lambda_1 e^{-\lambda_1 x_1}, \quad x_1 \geq 0,$$

and

$$f_{X_2|X_1}(x_2|x_1) = (\lambda_2 + \alpha x_1)e^{-(\lambda_2+\alpha x_1)x_2}, \quad x_2 \geq 0.$$

The first variable, X_1, is exponential and has the rate λ_1, and the second variable, X_2, is exponential conditionally on X_1, and has a rate $\lambda_2(X_1) = \lambda_2 + \alpha X_1$. The parameter α determines the strength of dependence between X_1 and X_2. We apply the Multivariate Inverse Transform Sampling as follows:

- Simulate a random number U from the uniform distribution on $[0,1]$ and compute $X_1 = -\frac{1}{\lambda_1}\log(1-U)$, which is the inverse transform sampling for an exponential distribution.

- Independently simulate another random number V from the uniform distribution on $[0,1]$ and calculate X_2 using the inverse CDF conditioned on X_1, calculated as $X_2 = -\frac{1}{\lambda_2+\alpha X_1}\log(1-V)$.

□

3.2.4 Acceptance-Rejection Method for Continuous Variables

We want to generate random numbers from a known density f. We assume that it is difficult to generate directly from f, but we can find a density h that is easy to generate from and a normalizing constant c such that

$$\sup_x \frac{f(x)}{h(x)} \leq c\ .$$

Simulation Algorithm: Acceptance-Rejection Method

- Generate a random number y from the density h.
- Independently, generate a random number u from the standard uniform distribution.
- If $u \leq \frac{f(y)}{ch(y)}$, then set $x = y$.
- Otherwise, go back to the first step.
- The value x is then a random number of the density f.

The function h is called the envelope function.

Theorem 3.2.11. *The Acceptance-Rejection algorithm returns a random number from the density f.*

Proof. We need to show that the random variable generated X follows the density f. To do this, we find the conditional distribution of Y given that $U \leq \frac{f(Y)}{ch(Y)}$. We have

$$\mathbb{P}\left(Y \leq x \mid U \leq \frac{f(Y)}{ch(Y)}\right) = \frac{\mathbb{P}(Y \leq x \cap U \leq \frac{f(Y)}{ch(Y)})}{\mathbb{P}(U \leq \frac{f(Y)}{ch(Y)})}\ .$$

For the denominator, applying the total probability rule and using the independence of U and Y, we get

$$\mathbb{P}\left(U \leq \frac{f(Y)}{ch(Y)}\right) = \int_{\mathbb{R}} \mathbb{P}\left(U \leq \frac{f(y)}{ch(y)}\right) h(y)dy = \int_{\mathbb{R}} \frac{f(y)}{ch(y)} h(y)dy = \frac{1}{c}\int_{\mathbb{R}} f(y)dy = \frac{1}{c} \, .$$

Similarly, the numerator is

$$\int_{-\infty}^{x} \mathbb{P}\left(U \leq \frac{f(y)}{ch(y)}\right) h(y)dy = \int_{-\infty}^{x} \frac{f(y)}{ch(y)} h(y)dy = \frac{1}{c}\int_{-\infty}^{x} f(y)dy = \frac{1}{c}F(x) \, .$$

Thus,

$$\mathbb{P}\left(Y \leq x \mid U \leq \frac{f(Y)}{ch(Y)}\right) = F(x) \, ,$$

as required.

From the proof, we also see that the overall acceptance probability is:

$$\frac{1}{c} \, .$$

□

Algorithm Complexity

- For each sample of size n, we need to generate an additional uniform random number for the acceptance step and perform a comparison. If the acceptance probability is $\frac{1}{c}$, the expected total number of trials will be cn.
- Let m_2 be the complexity of generating a single candidate sample from the envelope distribution $h(x)$.
- Therefore, generating cn candidate samples using the acceptance-rejection method takes $O(c\, m_2\, n)$ operations.

Example 3.2.12. To generate a random variable with a normal distribution with parameters $\mu = 1$ and $\sigma = 1$, we consider the probability density function (PDF) given by:

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \, .$$

The corresponding cumulative distribution function (CDF) is:

$$F(x;\mu,\sigma) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \, ,$$

where $\operatorname{erf}(x)$ is the error function defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,dt\ .$$

Due to the complexity of the error function $\operatorname{erf}(x)$, there is no closed-form inverse for the CDF of the normal distribution. Therefore, we apply the acceptance-rejection method. Since the normal distribution X is symmetric around its mean, we can generate $|X|$ first and then apply a random sign to the generated value to obtain a sample from the normal distribution.

We choose the envelope function as $h(x) = e^{-x}$, which is the standard exponential density. It can be shown that:

$$\frac{\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}}{e^{-x}} = \frac{1}{\sqrt{2\pi}}e^{\frac{x-x^2}{2}} \leq \frac{e}{\sqrt{2\pi}} =: c\ ,$$

since $x - \frac{x^2}{2}$ achieves its maximum at $x = 1$.

The steps of the algorithm are as follows:

- Generate a candidate y from the exponential distribution $h(y)$ with CDF $H(y) = 1 - e^{-y}$ and inverse CDF $H^{-1}(u) = -\ln(1-u)$.
- Generate a uniform random number u_1 on $[0, 1]$.
- Calculate $y = -\ln(1 - u_1)$.
- Generate another uniform random number u_2 on $[0, 1]$.
- If $u_2 \leq \frac{e^{y-y^2/2}}{c}$, set $|x| = y$.
- Generate another uniform random number u_3 on $[0, 1]$.
- If $u_3 \leq 0.5$, set $x = -|x|$. Otherwise, set $x = |x|$.

□

3.2.5 Transformations

Problem Setup. We want to generate random numbers from a complex, known distribution F. Transformation techniques allow us to generate a target distribution when a random variable X with distribution F depends on another random variable. This is a generalization of the previously introduced quantile transform method.

- Assume we can simulate from a random vector Y with values in $\mathbb{R}^p$. For example, Y could be a vector of independent uniform random variables.

- Let $\phi : \mathbb{R}^p \to \mathbb{R}$ be a measurable function.
- Define $X = \phi(Y)$.

Simulation Algorithm.

- Simulate a random variable Y from its distribution using the inverse transformation method. This involves generating a uniform random variable U and then applying $Y = F_Y^{-1}(U)$. Denote these samples as y_j, $j = 1, \dots, n$.
- Apply the function ϕ to each sample y_j to obtain $x_j = \phi(y_j)$.

Algorithm Complexity.

- Assume the complexity of generating a sample of size n from the distribution Y is m_y.
- The complexity of applying the function ϕ to each of the n samples is $O(n \cdot m_\phi)$, where m_ϕ denotes the complexity of applying ϕ to a single sample.
- Therefore, the total complexity of the algorithm is given by $O(m_y + n \cdot m_\phi)$.

Example 3.2.13. To simulate a random variable from the chi-square distribution with k degrees of freedom, we can sum the squares of k independent standard normal random variables. Let $Z_1, Z_2, \dots, Z_k$ be independent standard normal random variables.

The transformation function $\phi(Z_1, Z_2, \dots, Z_k)$ that gives a chi-square distributed random variable X is:

$$\phi(Z_1, Z_2, \dots, Z_k) = Z_1^2 + Z_2^2 + \cdots + Z_k^2 .$$

Thus, the random variable $X = \phi(Z_1, Z_2, \dots, Z_k)$ follows a chi-square distribution with k degrees of freedom. $\square$

3.2.6 Conditioning and Mixture Distributions

This is a generalization of the previously introduced acceptance-rejection method. To generate a random sample from the conditional distribution $F_Y(\cdot \mid c)$ of a random variable Y given a constraint function c, the following steps can be used:

- For a constraint function c that involves only Y:
  - Simulate a sample y according to the distribution of Y.
  - Evaluate the constraint function $c(y)$.

  - If $c(y)$ satisfies the constraint, keep the sample y; otherwise, discard it and repeat the process.

- For a constraint c that involves multiple random variables, denote these variables as X, where X could be a vector of random variables:
  - Simulate a pair (X, Y) from their joint distribution.
  - Evaluate the constraint function $c(X, Y)$.
  - If $c(X, Y)$ satisfies the constraint, keep the sample Y; otherwise, discard it and repeat the process.

Algorithm Complexity

- Each generation of a sample y from the distribution of Y or a pair (X, Y) from their joint distribution is considered one operation.
- Evaluating the constraint function $c(y)$ for a single variable or $c(X, Y)$ for multiple variables is also considered one operation per evaluation.
- The expected number of iterations (generation and evaluation) needed depends on the probability that a generated sample satisfies the constraint. Each trial involves generating a sample (which is one operation) and evaluating the constraint (which is another operation). Therefore, each trial takes approximately 2 operations. If the probability is p for a constraint involving only Y, the expected number of operations is approximately $\frac{2}{p}$. For a constraint involving (X, Y), if the probability is q, the expected number of operations is approximately $\frac{2}{q}$.

Thus, if the probability of y satisfying $c(Y)$ or (X, Y) satisfying $c(X, Y)$ is p or q, respectively, the total number of operations needed to generate a valid sample Y is approximately $\frac{2}{p}$ when the constraint involves only Y and $\frac{2}{q}$ when the constraint involves both X and Y.

Example 3.2.14. Suppose that we want to generate a random variable X that follows the $\beta(a, b)$ (Beta) distribution using two independent random variables U and V, both uniformly distributed on $[0, 1]$. We apply a transformation and a conditioning step as follows:

- **Transformation and Condition:**
  - Simulate U and V from the uniform distribution on $[0, 1]$.
  - Compute the transformations $U^{1/a}$and $V^{1/b}$.
  - Check the condition $U^{1/a} + V^{1/b} \leq 1$.

  - If the condition is satisfied, compute X as
$$X = \frac{U^{1/a}}{U^{1/a} + V^{1/b}}.$$
  - If the condition is not met, repeat the simulation from step 1.

- **Proof:** We prove that this algorithm produces a random number from the required Beta distribution. We first note that
$$\begin{aligned}
\mathbb{P}\left(U^{1/a} + V^{1/b} < 1\right) &= \int_0^1 \int_0^1 1\left\{u^{1/a} + v^{1/b} < 1\right\} \mathrm{d}u \ \mathrm{d}v \\
&= \int_0^1 \left(1 - v^{1/b}\right)^a \ \mathrm{d}v = b\int_0^1 t^{b-1}(1-t)^a \ \mathrm{d}t \quad \left(t = v^{1/b}\right) \\
&= \frac{b\Gamma(b)\Gamma(a+1)}{\Gamma(a+b+1)} = \frac{\Gamma(b+1)\Gamma(a+1)}{\Gamma(a+b+1)}.
\end{aligned}$$
Thus, for any bounded function g,
$$\begin{aligned}
\mathbb{E}[g(X)] &= \frac{\Gamma(a+b+1)}{\Gamma(b+1)\Gamma(a+1)} \int_0^1 \int_0^1 g\left(u^{1/a}/\left(u^{1/a} + v^{1/b}\right)\right) 1\left\{u^{1/a} + v^{1/b} < 1\right\} \mathrm{d}u \ \mathrm{d}v \\
&= \frac{\Gamma(a+b+1)}{\Gamma(b+1)\Gamma(a+1)} \int_0^1 \int_0^1 g(x)abx^{a-1}(1-x)^{b-1}y^{a+b-1} \ \mathrm{d}x \ \mathrm{d}y.
\end{aligned}$$
The change of variables $u = (xy)^a, v = ((1-x)y)^b$ gives
$$\begin{aligned}
&\frac{\Gamma(a+b+1)}{\Gamma(b+1)\Gamma(a+1)(a+b)} \int_0^1 g(x)abx^{a-1}(1-x)^{b-1} \ \mathrm{d}x \\
&= \frac{\Gamma(a+b)}{\Gamma(b)\Gamma(a))} \int_0^1 g(x)x^{a-1}(1-x)^{b-1} \ \mathrm{d}x.
\end{aligned}$$
This shows that X has a $\beta(a,b)$ distribution.

$\square$

3.2.7 Simulating from Parametric Families

We assume that our distribution has the form $F(\cdot;\theta)$, where F is known, but the parameter $\theta \in \mathbb{R}^d$ is unknown.

Simulation Algorithm

- Estimate the parameter θ for the model based on the given data.
- Then use one of the methods mentioned in the previous sections to generate data from the estimated distribution.

Example 3.2.15. Suppose that we have a dataset and we assume that the data follow a normal distribution with mean μ and variance σ^2, denoted by $\mathcal{N}(\mu, \sigma^2)$.

- First, we estimate the parameters μ and σ^2 using the sample mean $\hat{\mu}$ and sample variance $\hat{\sigma}^2$.
- Then, we generate data from the distribution $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ using the inverse transformation method.

□

3.2.8 Generating Multivariate Distribution with Decomposition-Based Method

To generate a multivariate vector $\mathbf{Y} = (Y_1, \ldots, Y_m)$ with dependent components, follow this approach: If the transformed vector $\mathbf{X} = L\mathbf{Y} + \mu$ retains the same type of distribution as the independent components $\mathbf{Y}$, where L is a transformation matrix and μ is a vector that adjusts the mean, then employ this method. We must ensure that μ and the covariance matrix $\Sigma = LL^T$ are properly defined to establish mean and variance.

Cholesky Decomposition

Theorem 3.2.16. *Let $\mathbf{Y}$ be a random vector in $\mathbb{R}^m$ with independent components and mean vector μ, let L be a deterministic matrix of size $m \times m$. Then $\mathbf{X} = L\mathbf{Y} + \mu'$ has a multivariate distribution with the mean vector $L\mu + \mu' \in \mathbb{R}^m$ and the covariance matrix $\Sigma = LL^T$.*

Proof.

- *Mean of* $\mathbf{X}$*:*

$$\mathbb{E}[\mathbf{X}] = \mathbb{E}[L\mathbf{Y} + \mu'] = L\mathbb{E}[\mathbf{Y}] + \mu' = L\mu + \mu'.$$

  Hence, the mean vector of $\mathbf{X}$ is $L\mu + \mu'$.

- *Covariance Matrix of* $\mathbf{X}$*:*

$$\begin{aligned}\Sigma = \mathrm{Cov}(\mathbf{X}) &= \mathbb{E}[(\mathbf{X} - \mathbb{E}[\mathbf{X}])(\mathbf{X} - \mathbb{E}[\mathbf{X}])^T] \\ &= \mathbb{E}[L(\mathbf{Y} - \mu)(\mathbf{Y} - \mu)^T L^T] \\ &= L\mathbb{E}[(\mathbf{Y} - \mu)(\mathbf{Y} - \mu)^T]L^T.\end{aligned}$$

  By the given conditions, the covariance matrix of $\mathbf{Y}$ is the identity matrix I:

$$\mathbb{E}[(\mathbf{Y} - \mu)(\mathbf{Y} - \mu)^T] = I.$$

  Substituting it into the expression for Σ:

$$\Sigma = LIL^T = LL^T.$$

Thus, $\mathbf{X}$ has mean vector $L\mu + \mu'$ and covariance matrix $\Sigma = LL^T$. Based on this theorem, we have the following algorithm that utilizes the Cholesky decomposition. □

Simulation Algorithm

(a) For given Σ, apply the Cholesky method to get L.

(b) Generate an m-dimensional vector $\mathbf{Y}$ with independent standard normal components.

(c) Calculate $\mathbf{X} = L\mathbf{Y} + \mu'$.

(d) Repeat (b)-(c) n times. As a result, we obtain a sample of size n from a multivariate normal distribution with the covariance matrix $\Sigma = LL^T$.

Example 3.2.17 (Simulation from Bivariate Normal)**.** Suppose that we want to generate samples from a bivariate normal distribution with mean vector $\mu = [0, 1]^T$ and covariance matrix

$$\Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}.$$

- The Cholesky decomposition of Σ is:

$$L = \begin{bmatrix} 1 & 0 \\ 0.7 & \sqrt{0.51} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0.7 & 0.714 \end{bmatrix}.$$

- Transform standard normal samples (y_1, y_2) using the Cholesky factor L:

$$(x_1, x_2)^T = L\left[y_1, y_2\right] = \begin{bmatrix} 1 & 0 \\ 0.7 & 0.714 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ 0.7y_1 + 0.714y_2 \end{bmatrix}.$$

- Add the mean vector μ to the transformed samples $\mathbf{X}$:

$$\mathbf{Z} = \mathbf{X} + \mu = \left[y_1, 0.7y_1 + 0.714y_2\right] + \left[10\right] = \begin{bmatrix} y_1 + 1 \\ 0.7y_1 + 0.714y_2 \end{bmatrix}.$$

This algorithm generates samples from a bivariate normal distribution with mean μ and covariance matrix Σ. □

Spectral Decomposition

Theorem 3.2.18. • *Any covariance matrix Σ can be factorized as $\Sigma = Q\Lambda Q^T$, where*

– *Λ is the diagonal matrix whose diagonal elements are the eigenvalues of Σ,*

  – *Q is an orthogonal matrix whose columns are the eigenvectors of Σ.*

- *Let $\mathbf{Y}$ be a random vector in $\mathbb{R}^m$ with mean vector 0 and identity covariance matrix. If $\mathbf{X} = Q\Lambda^{\frac{1}{2}}\mathbf{Y}$, then $\mathbf{X}$ has a covariance matrix Σ.*

Proof. Given that $\mathbf{Y}$ has the mean vector 0, we note that

$$\begin{aligned}\mathrm{Var}(\mathbf{X}) &= \mathbb{E}[\mathbf{X}\mathbf{X}^T] = \mathbb{E}\left[(Q\Lambda^{\frac{1}{2}}\mathbf{Y})(Q\Lambda^{\frac{1}{2}}\mathbf{Y})^T\right] \\ &= \mathbb{E}\left[Q\Lambda^{\frac{1}{2}}\mathbf{Y}\mathbf{Y}^T\Lambda^{\frac{1}{2}}Q^T\right] = Q\Lambda^{\frac{1}{2}}\mathbb{E}[\mathbf{Y}\mathbf{Y}^T]\Lambda^{\frac{1}{2}}Q^T \\ &= Q\Lambda Q^T = \Sigma\end{aligned}$$

which is the desired covariance matrix. □

Simulation Algorithm

(a) For a given Σ, apply the spectral decomposition to get Q.

(b) Generate an m-dimensional vector $\mathbf{Y}$ with independent standard normal components.

(c) Calculate $\mathbf{X} = Q\Lambda^{1/2}\mathbf{Y}$.

(d) Repeat (b)-(c) n times. As a result, we obtain a sample of size n from a multivariate normal distribution with the covariance matrix $\Sigma = Q\Lambda Q^T$.

Example 3.2.19. Suppose we want to generate normal data with the following parameters:

- Mean vector: $\mu = \begin{bmatrix}1\\0\end{bmatrix}$.

- Covariance matrix: $\Sigma = \begin{bmatrix}1 & 0.7\\0.7 & 1\end{bmatrix}$.

We proceed as follows:

- The spectral decomposition of Σ is: $\Sigma = Q\Lambda Q^T$, where

$$Q = \begin{bmatrix}-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}, \quad \Lambda = \begin{bmatrix}1.7 & 0\\0 & 0.3\end{bmatrix}.$$

- Generate standard normal samples Y:

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix},$$

  where y_1 and y_2 are realizations of independent standard normal variables.

- Transform the samples using Q and $\Lambda^{1/2}$:

$$\mathbf{X} = Q\Lambda^{1/2}Y = \begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \sqrt{1.7} & 0 \\ 0 & \sqrt{0.3} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{3.4}}{2}y_1 + \frac{\sqrt{0.6}}{2}y_2 \\ \frac{\sqrt{3.4}}{2}y_1 + \frac{\sqrt{0.6}}{2}y_2 \end{bmatrix}.$$

The resulting data $\mathbf{Z}$ follows a multivariate normal distribution with mean vector $\mu = [1, 0]$ and the covariance matrix Σ. □

3.2.9 Bootstrap

If we do not have a model assumption, we can apply non-parametric data generation techniques. Our objective is to randomly choose individuals from a dataset $\mathcal{D} = \{X_1, \ldots, X_n\}$. Each individual should have the same probability of being selected, and we repeat the selection process N times.

Simulation Algorithm

- We generate independently N values $U = \{u_1, \ldots, u_N\}$ from the standard uniform distribution.
- for each value $u_j \in U$, $j = 1, \ldots, N$, if u_j is between $\frac{i-1}{n}$ and $\frac{i}{n}$ for some $i = 1, \ldots, n$, then set $x_j = X_i$.
- Note that some x_j's may have the same value X_i.

3.2.10 Equivalence of inverse transformation and bootstrap

Inverse transformation method applied to the empirical cumulative distribution function (ECDF) is equivalent to the bootstrap procedure.

Let $r(x)$ denote the number of times a value x appears in the dataset:

$$r(x) = \sum_{j=1}^{N} 1\{X_j = x\}.$$

The empirical CDF at the point x is:

$$\widehat{F}_n(x) = \frac{1}{n}\sum_{j=1}^{N} 1\{X_j \leq x\}.$$

Define x^- as the largest value less than x in the dataset (hence, both x^- and x are random numbers). The probability that u falls within this interval, thus selecting x by the inverse CDF, is:

$$\mathbb{P}\left(u \in \left(\widehat{F}_n(x^-), \widehat{F}_n(x)\right]\right) = \frac{r(x)}{n}.$$

This probability is the same as the bootstrap method in which each data point is equally likely to be chosen.

3.2.11 SMOTE

SMOTE stands for Synthetic Minority Oversampling Technique; [4]. This method was popular at some point with practitioners working in data privacy or data generation in general. A big drawback of this method is that there is no associated mathematical theory. Another drawback is the emergence of newer data generation methods, such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and diffusion models (which lack the mathematical theory as well ...). These methods are now more commonly used because they can generate more realistic and varied synthetic data. We mention the SMOTE method, since it is still implemented in some packages.

We start with some terminology. A class refers to the categories, groups, or bins in which data points are grouped based on characteristics. It shares a similar idea from the equivalence classes previously discussed. A minority class refers to the class that has fewer samples compared to other classes. For example, in a demographic study, ages could be categorized as classes, with individuals over 90 forming a minority class because of their fewer numbers.
Balancing a data set involves adjusting the proportions of different classes to prevent biases in machine learning models caused by unequal class distributions. This can be achieved by oversampling the minority class - increasing its representation by duplicating existing samples or generating new synthetic ones - or by undersampling the majority class, where excess samples are removed to equalize the class sizes. Note that balancing dataset is optional in the `SMOTE` algorithm.

In what follow we have:

- T_j is the set of data points classified under the j-th minority class before any adjustments.
- t_j is a subset of selected data points in T_j, and this subset is chosen to modify the class size to achieve a more representative distribution between different classes.

- N_j is the percentage of the total data points in T_j that are included in the subset t_j. It determines the proportion of the minority class used in the balance process.

- K denotes the number of nearest neighbors to consider in the given algorithm.

Simulation Algorithm. We can divide the algorithm into two stages: the first stage involves balancing the dataset, and the second stage focuses on the generation of synthetic data.

- Determine the proportion of the minority class T_j, the desired balance N_j, and the maximum number of nearest neighbors K.
- For the j-th minority class T_j, choose a subset denoted by t_j such that N_j percent of the sample from T_j is selected.
- Randomly choose a data point x_i from the balanced dataset t_j.
- Randomly select a number k_i from the set $\{1, \ldots, K\}$.
- Identify the k_i-th nearest neighbor of x_i, denoted $x_i^{k_i}$.
- Randomly generate a number u between 0 and 1.
- Generate synthetic data $x_{i,new} = x_i + u(x_i^{k_i} - x_i)$.

We note that the nearest neighbour may not be unique. This requires a special treatment.

Example 3.2.20. In our simulation, we generate 10 data points from a multivariate normal distribution and categorize them into two minority classes, T_1 and T_2. We selected $N_j = 100\%$ of the data points in each class, eliminating the need for balancing. The parameter k is set to 2, indicating that we will use the two nearest neighbors of each selected data point to generate synthetic data points. This approach ensures that each class is adequately represented while maintaining the statistical characteristics of the original dataset.

x_1	x_2
0.49671	−0.13826
0.64769	0.77997
−0.23415	−0.23414
1.57921	1.54256
0.76743	0.96940
−0.46947	0.54256
−0.46342	−0.46573
0.24196	−0.04535
−1.91328	−1.72492
0.56229	1.31425

Suppose we selected point $(0.49671, -0.13826)$ and two nearest neighbors (according to the Euclidean distance) of the point are $(0.64769, 0.77997)$ and $(0.76743, 0.96940)$. We will generate synthetic data points by interpolating these points.

1. Randomly select k_i from the set $\{1, 2\}$. Assume $k_i = 2$ is chosen for this example.
2. The second nearest neighbor selected according to k_i is $(0.76743, 0.96940)$.
3. Choose a random interpolation factor $u = 0.65$.
4. Calculate the synthetic point:

$$\begin{aligned} x_{\text{new}} &= (0.49671, -0.13826) + 0.65 \times ((0.76743, 0.96940) - (0.49671, -0.13826)) \\ &= (0.67276, 0.58526). \end{aligned}$$

Same procedure will apply to each data point in the original dataset. In this example, we have demonstrated the detailed steps for generating synthetic data using the SMOTE method. □

3.2.12 Linear interpolation techniques

We aim to create synthetic data from an observed dataset. Although the underlying distribution of the samples is continuous, we have access to only discrete samples from these data. The following method is designed to address this issue. In what follow we have:

- n'_d represents the number of distinct values in dimension d, where $n'_d \leq n$, and n is the total number of data points. In the one-dimensional case, we write $n'_1 = n' = n$.
- For each d-dimensional data point $(x_{j,1}, \ldots, x_{j,d})$, $j = 1 \ldots, n$ we read out the ith coordinates and we order them:

$$x_{(1),i} \leq x_{(2),i} \leq \cdots \leq x_{(n'),i}.$$

- Interval Definition: Based on the ordered values, we create intervals in each dimension $i = 1, \ldots, d$:

$$\underbrace{(x_{(1),i}, x_{(2),i}]}_{=I_{1,i}}, \ldots, \underbrace{(x_{(\ell),i}, x_{(\ell+1),i}]}_{=I_{\ell,i}} \ldots \underbrace{(x_{(n'-1),i}, x_{(n'),i}]}_{=I_{n'-1,i}}.$$

  This defines the interval bounds between consecutive values in the sorted data for each dimension.

- The d-dimensional interval is the Cartesian product of intervals across all dimensions:

$$I'_\ell = I_{\ell,1} \cdots \times \cdots \times I_{\ell,d}, \quad \ell = 1, \ldots, n' - 1. \tag{3.1}$$

  This defines multidimensional intervals.

Below, we outline the algorithm for 1-dimensional case only ($d = 1$). Hence, we drop the second subscript from x.

Linear Interpolation with additional point I

1. Sort the data points by their values, $x_{(1)}, \ldots, x_{(n')}$.
2. Add an additional point $x_{(0)}$ that is smaller than $x_{(1)}$ to adjust the interpolation.
3. Randomly select m points from $x_1, \ldots, x_n$.
4. For each selected data point $x_{(\ell)}$, consider the interval $(x_{(\ell-1)}, x_{(\ell)})$.
5. Draw a uniform sample from the interval.

The outlined algorithm is equivalent to applying the inverse transformation method to the piecewise linear function that interpolates between each point on the empirical CDF. The reason for interpolating to the left is the natural right-continuity of the CDF function.
We notice that:

- The linear interpolation algorithm does not naturally apply any smoothing for values smaller than the smallest x in the data. This means that the tail distribution needs to be estimated separately.

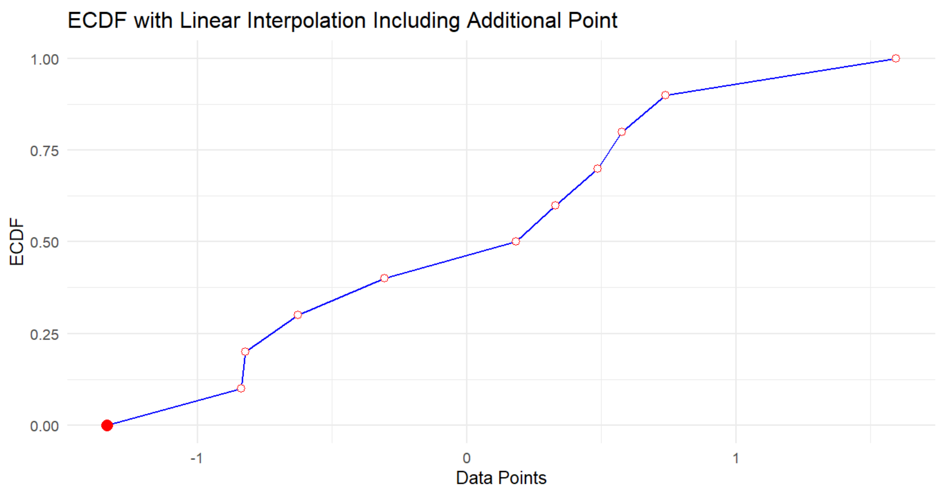


Figure 3.1: Linear Interpolation with additional point I

- Moreover, the support of dataset also cannot be guaranteed, that is, the lower bound of the data is smaller than the original data and the upper bound of the data does not exceed the support.

The following plot shows the ECDF of the original data, with blue lines representing the cumulative probability using linear interpolation. An additional red point is added below the minimum value with an ECDF of 0. This added additional point creates some bias and boundary problems. An alternative method to solve this problem is as follows (we call it Linear Interpolation II). Instead of adding an additional point at the smallest value of x, we assign values generated between $x_{(1)}$ and $x_{(2)}$ a higher probability. This approach not only avoids the need for tail estimation, but also preserves the support of the dataset (see Figure 3.2).

Multivariate case. In the multivariate case, the empirical cumulative distribution function (ECDF) and linear interpolation functions become more complex. For ECDF, it no longer calculates cumulative probabilities for one variable but rather understands how multiple variables interact collectively. Moreover, linear piecewise functions in these spaces involve fitting high-dimensional planes to data segments, rather than just connecting points. As shown in Section 3.2.10 we can use bootstrap as an alternative to inverse transformation methods to sample the distribution. This becomes a smoothing bootstrap problem.
We extend our Linear Interpolation II algorithm to the multi-dimensional case. Instead of sampling uniformly over intervals in dimension one, we sample uniformly over the boxes I_ℓ defined in (3.1).

Example 3.2.21. We generate 1000 samples from a multivariate normal distribution

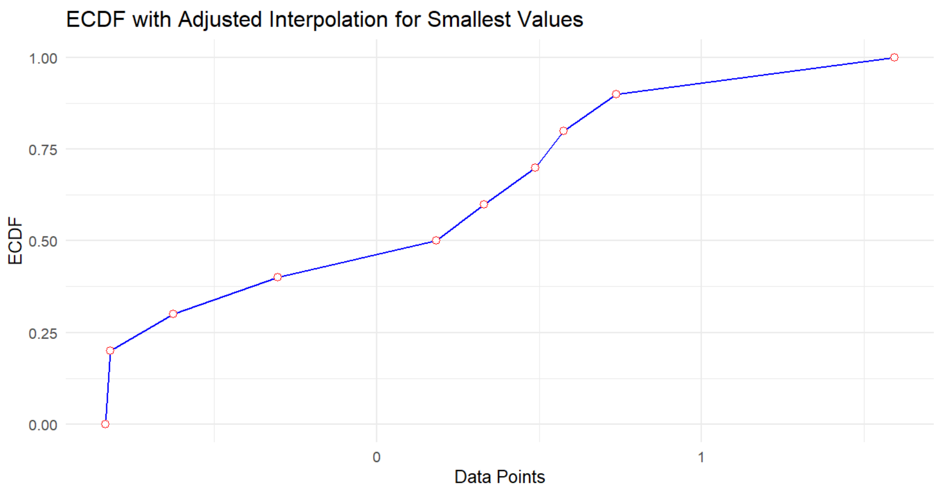


Figure 3.2: Linear Interpolation II

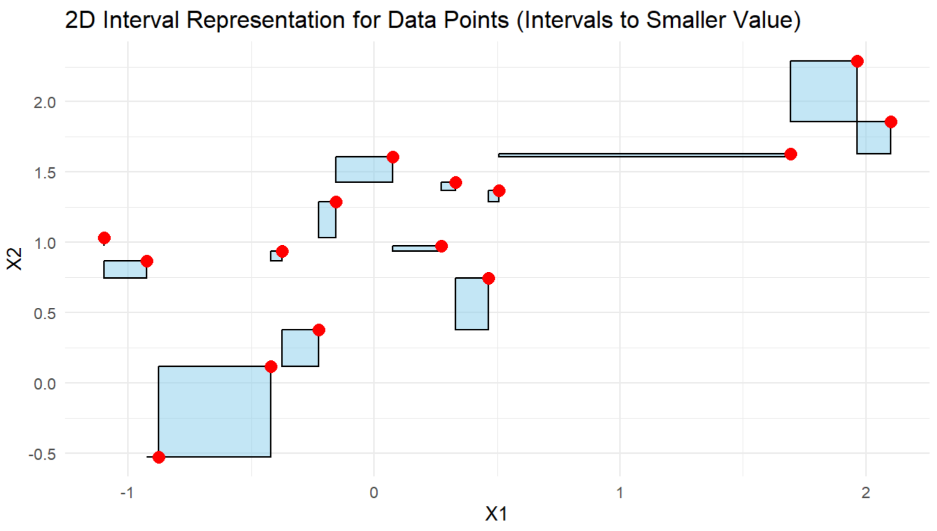
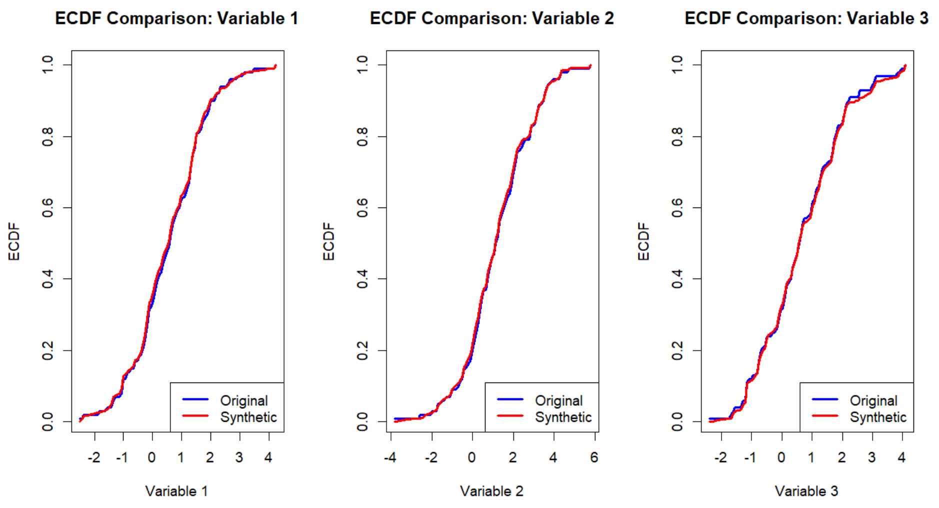
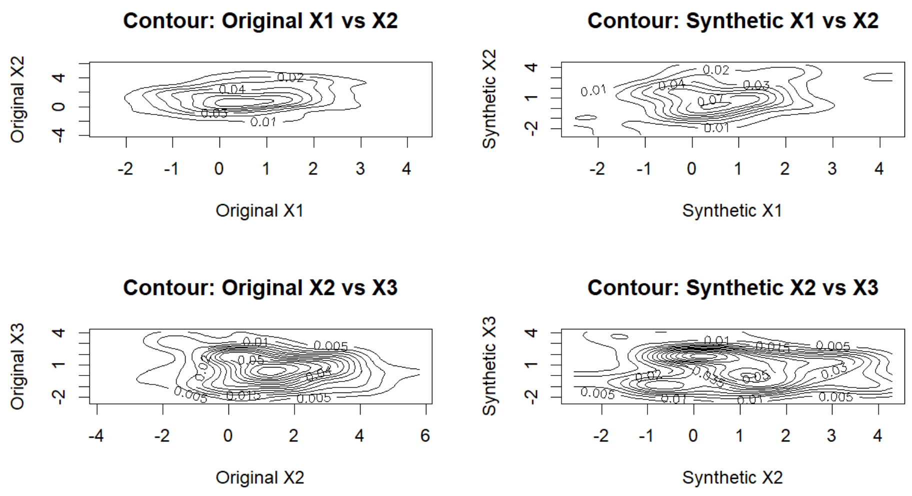


Figure 3.3: Linear interpolation II in dimension 2.

and compare the ECDFs for the marginal distributions and the dependence structure using contour plots.





The synthetic data follows the same marginal distribution and dependence structure as the original ones. However, we can observe some distortion to the left tail of the original ECDF. As the number of original data points increases, this issue becomes negligible. □

We make the following comments:

- Compared to other methods, these generation techniques ensure maximum utility with respect to the empirical cumulative distribution function (CDF).

- For each interpolation method, we have the flexibility to sample not only uniformly but also from other distributions. For example, in the Linear Interpolation II algorithm, we can modify Step 5 to draw a normal sample from the interval. Different distributions can be chosen for various purposes. For example, for privacy purposes, we might want to generate synthetic data that deviates significantly from the original data. In such cases, the chosen distribution would focus more on the center of the interval, such as a normal distribution or distributions with lighter tails.

- Different intervals can be selected, and various sampling methods can be applied within each interval. This transforms the algorithm into a problem similar to bootstrap smoothing [2] [15] or adaptive bandwidth kernel density estimation [21] [35] [1].

Chapter 4

Synthpop Package: Description, Challenges and Solutions

The synthpop package for R is a tool specifically designed for generating synthetic datasets from the original dataset, trying to keep the statistical properties of individual-level data. The synthpop package for R is based on the synthetic data generation techniques developed in several foundational papers, including those in [25], [18], [22], and [23]. The methodology used in these articles is based on the concept of multiple imputation, originally designed to handle missing data, as Rubin detailed in [26]. This approach has been adapted in the synthpop package to facilitate the generation of synthetic datasets that allow the analysis of sensitive datasets without compromising individual privacy.

In our analysis of the synthpop package, we focus particularly on how effectively it uses data utility metrics for the generated data, highlighting any limitations or challenges. We start by reviewing different data utility metrics. We discuss their limitations and potential lack of interpretation. This is illustrated by many examples. We indicate that some of the statistical inference tools implemented in the package are not used correctly. For example, the famous Kolomogorov-Smirnov test is applied to compare the original and synthetic datasets. The null hypothesis of equality between two datasets is accepted or rejected based on the calculated p-values. As we will show and discuss, this is incorrect since the datasets are dependent. These topics are discussed in Section 4.4.

Next, we discuss potential issues with implementation of data generation procedure. If our goal is to generate synthetic data from a multivariate distribution, we need to provide a proper sequence in which the variables are generated. In other words, we would like to preserve not only the univariate characteristics of the original dataset, but also the dependence structure between the variables. The package does not allow to verify if the dependence structure is well-preserved. Furthermore, if there is no sequential relation between the variables in the original dataset, the package will not generate the proper synthetic dataset. These topics are discussed in Section 4.5.

We propose a solution to this problem. Rather than sequentially fitting models to the data, which might lead to compounding errors and potential privacy leaks, we propose a method that involves fitting each variable independently, using all other variables in the data set as predictors.

Finally, the package does not discuss privacy of the synthetic data, focusing on data utility. We will look at the privacy of synthetic data using the notion of differential privacy introduced before. These topics are discussed in Section 4.6.

4.1 Terminology

- We have observations $(X_{1j}, \ldots, X_{pj})$, where $j = 1, \ldots, n$, drawn from a multivariate vector $(X_1, \ldots, X_p)$. When necessary, we refer to these observations as $(X_{1,\text{obs}}, \ldots, X_{p,\text{obs}})$. Thus, the observed data can be written as $(X_{1j,\text{obs}}, \ldots, X_{pj,\text{obs}})$, for $j = 1, \ldots, n$.

- We aim to generate synthetic data $(X_{1,\text{syn}}, \ldots, X_{p,\text{syn}})$, which should ideally retain the statistical properties of the original data $(X_{1,\text{obs}}, \ldots, X_{p,\text{obs}})$.

- The function $f(X_{i+1,\text{obs}} \mid X_{i,\text{obs}}, \ldots, X_{1,\text{obs}})$ represents the conditional distribution of $X_{i+1,\text{obs}}$ given all previously observed variables. The choice of the model for f is versatile and can depend on the nature of the data.

- The visit sequence is an ordered list that determines the sequence in which the variables are synthesized, ensuring that the dependencies between the variables are respected in the synthetic data set: $X_{1'}, \ldots, X_{p'}$. Note that, e.g. $X_{1'}$ is not necessary $X_{1,\text{syn}}$.

- The predictor matrix $\mathbf{P}$ is a square matrix with dimensions $p \times p$, where p is the number of variables in the dataset. Each row and column of the matrix correspond to one of the variables $X_1, X_2, \ldots, X_p$. Each entry P_{ij} in the matrix is binary: $P_{ij} = 1$ indicates that variable X_j is used as a predictor of variable X_i, while $P_{ij} = 0$ indicates that variable X_j is not used as a predictor of variable X_i. The package allows the input of a predictor matrix that represents the correlations or dependencies between variables, it also outputs a predictor matrix that reflects the correlations used by the package during data generation. It is important to note that the input predictor matrix may differ from the output predictor matrix, as will be demonstrated in example 4.2.2.

- Model fitted for the $(i+1)'$-th variable in the visiting sequence is denoted by $f\left(X_{(i+1)'} \mid \mathbf{X}_t\right)$, where the set of predictors $\mathbf{X}_t$ is composed of those variables from the set $(X_{1'}, X_{2'}, \ldots, X_{i'})$ that are indicated by the input predictor matrix $\mathbf{P}$. We denote $P_{(i+1)'}$ as the $(i+1)'$-th row of the predictor matrix $\mathbf{P}$.

4.2 Simulation algorithm

The following simulation algorithm is designed to generate synthetic data by sequentially modeling each variable in the dataset based on the previously synthesized variables. Ideally, this method ensures that the dependencies between variables are maintained, reflecting the structure present in the original data. However, we will demonstrate that this is not always the case if we use the package. As such, we propose some solutions to deal with this issue.

The process begins by selecting an initial variable from the observed data, generating its synthetic counterpart by e.g. bootstrapping, and then progressively modeling and generating synthetic values for each subsequent variable using a fitted model. The approach is iterative, with each synthetic variable conditioned on all previously generated synthetic variables. Note that the sequence of selecting of variables depend on the input visiting sequence.

Simulation Algorithm

- According to the visit sequence denoted by $X_{1'}, \ldots, X_{p'}$, select the initial variable $X_{1',\text{obs}}$. Generate $X_{1,\text{syn}}$ by random sampling with replacement from $X_{1',\text{obs}}$.
- For each subsequent variable $X_{(i+1)',\text{obs}}$ in the visit sequence, select a subset $\mathbf{X}_t$ of variables from $X_{1'}, X_{2'}, \ldots, X_{i'}$ according to the input predictor matrix $\mathbf{P}$, and fit a model $f(X_{(i+1)',\text{obs}} \mid \mathbf{X}_t)$. The predictor matrix specifies which variables among $X_{i'}, X_{(i-1)'}, \ldots, X_{1'}$ are used to predict $X_{(i+1)'}$. This model represents the conditional distribution of $X_{(i+1)'}$ given the selected predictors.
- Draw synthetic values $X_{(i+1)',\text{syn}}$ from the model $f(X_{(i+1)',\text{syn}} \mid \mathbf{X}_t)$.
- Repeat the process for all variables in the dataset.

We make the following comments:

- For each fitted model, specific model assumptions are necessary. For continuous data, linear regression or CART may be used, while for categorical data, models such as logistic regression or regression trees may be appropriate.
- The choice of the initial variable and the sequence of selection variables must be specified.
- The predictor matrix that indicates the relationships between the variables should be specified.

- Random noise can be added to each predicted variable. There are two methods to incorporate this noise: one method is to add random noise directly to the predicted model, such as $X_{(i+1)',\text{syn}} = \hat{\beta}_i X_{i',\text{syn}} + \cdots + \hat{\beta}_1 X_{1',\text{syn}} + \epsilon$; where $\hat{\beta}_1, \ldots, \hat{\beta}_i$ are estimates of the original model. Alternatively, a Bayesian approach can be used to estimate the distribution of β, from which samples are drawn, assuming normal noise. For example, this method would use $X_{(i+1)',\text{syn}} = \tilde{\beta} X_{i',\text{syn}}$, where $\tilde{\beta} \sim N(\hat{\beta}, \sigma(\hat{\beta}))$.

The method used by `synthpop` involves generating synthetic data by sequentially modeling each variable based on previously synthesized variables. The algorithm introduced above is used, and the models can be chosen from various options. We provide a description of several commands from the package, as the description provided in the manual does not fully reflect what the functions actually do.

In what follows:

- y: Represents $X_{(i+1)',\text{obs}}$ when fitting the model.
- $\hat{y}$: Represents $X_{(i+1)',\text{syn}}$ when generating synthetic data.
- x: Represents a column vector $(X_{1',\text{obs}}, \ldots, X_{i',\text{obs}})^T$, used when fitting the model.
- Subset x_t: A selection of variables from the vector $x = (X_{1',\text{obs}}, \ldots, X_{i',\text{obs}})^T$. The selection is determined by the specifications of the predictor matrix $\mathbf{P}$.
- xp: Represents a column vector $(X_{1',\text{syn}}, \ldots, X_{i',\text{syn}})^T$, used when generating synthetic data.
- β: A column vector of parameters used in the models.
- $\hat{\beta}$ and $\tilde{\beta}$: Represent the estimated values of β using the classical and Bayesian approaches, respectively.

We have the following models provided by the package:

- The **.norm.fix.syn** function takes three main inputs,

$$y = X_{(i+1)',\text{obs}}, \quad x = (X_{1',\text{obs}}, \ldots, X_{i',\text{obs}})^T$$

  and predictor matrix $\mathbf{P}$. This function computes the regression coefficients $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_i)$ and estimates the variance of the error σ^2 for a linear regression model with regularization of the ridge:

$$y = x_t^T \beta + Z, \quad Z \sim N(0, \sigma^2).$$

  This model is designed to calculate the deterministic part of the response and then adds normally distributed noise to model errors or unexplained variation.

- The **.norm.draw.syn** function also requires three inputs

$$y = X_{i+1,\text{obs}}, \quad x = (X_{1',\text{obs}}, \ldots, X_{i',\text{obs}})^T$$

  and predictor matrix **P**. It uses Bayesian linear regression with ridge penalization to compute the regression coefficients β and σ^2:

$$y = x_t^T \tilde{\beta} + Z, \quad Z \sim N(0, \tilde{\sigma}^2).$$

  where x_t is a subset of variables selected from $x = (X_{1',\text{obs}}, \ldots, X_{i',\text{obs}})^T$, based on the input predictor matrix. Here, $\tilde{\beta}$ represents the regression coefficients adjusted for Bayesian inference, integrating prior belief (regularization) about the distribution of the coefficients with noise added to reflect uncertainty.

- The `syn.lognorm` function taking input variables y, x, xp and **P** which represent the observed response variable, observed predictors, synthetic predictors and predictor matrix, respectively. Note that y need to be positive. The predictors x_t are then used to fit a regression model either through a standard penalized linear approach (via `.norm.fix.syn`) or a Bayesian approach with ridge penalization (via `.norm.draw.syn`). Using estimated coefficients and error terms, the function generates synthetic values for the response variable:

$$\widehat{\log(y)} = (x_t p)^T \beta' + Z, \quad Z \sim N(0, \sigma^2),$$

  where β' are the estimated coefficients of $\hat{\beta}$ or $\tilde{\beta}$. Finally, the function reverses the logarithmic transformation by exponential $\widehat{\log(y)}$ to obtain $\hat{y}$.

- The `syn.sqrtnorm` function taking input variables y, x, xp and **P** representing the observed response variable, observed predictors, synthetic predictors and predictor matrix, respectively. We need to ensure that y contains only nonnegative values, since it undergoes a square root transformation, which is inappropriate for negative values. After transforming y by taking its square root, the predictors x are utilized to fit a regression model using a standard penalized linear approach (using `.norm.fix.syn`) or a Bayesian approach with ridge penalization (using `.norm.draw.syn`).

  Using the calculated coefficients and error terms, the function generates synthetic values for the transformed response variable:

$$\widehat{\sqrt{y}} = (x_t p)^T \beta' + Z, \quad Z \sim N(0, \sigma^2),$$

  where β' represents the coefficients estimated from either $\hat{\beta}$ or $\tilde{\beta}$. Subsequently, the function squares these values to reverse the transformation, obtaining the synthetic values for y.

- The `syn.normrank` function taking input variables y, x, xp and $\mathbf{P}$ which represent the observed response variable, observed predictors, synthetic predictors and predictor matrix, respectively. This function is adapted for regression synthesis using Z scores derived from the ranks of y. Initially, the function computes the Z scores for y using the quantile function (qnorm) applied to the ranks of y, normalized by the length of y plus one to avoid the extremes of the distribution. The predictors x_t are then used to fit a regression model through either a standard penalized linear method (via `.norm.fix.syn`) or a Bayesian approach with ridge penalization (via `.norm.draw.syn`). This fitting process aims to model the transformed scores rather than the original values, maintaining the distributional characteristics. Once the model is fitted, synthetic Z scores are predicted for the synthetic predictors xp, and these scores are transformed back to the data scale using the inverse cumulative distribution function (pnorm). The function ensures that the predicted ranks are within the valid range and then maps these ranks back to the corresponding values in y.

- The `syn.logreg` function taking input variables y, x, xp and $\mathbf{P}$ which represent the observed response variable, observed predictors, synthetic predictors and predictor matrix respectively. This function fits a logistic regression model to estimate the probability that y equals 1 given x. The logistic regression model is fitted using:

$$\log\left(\frac{p}{1-p}\right) = x_t^T\beta,$$

  where p is the probability of $y = 1$. The function allows the user to use fixed β as the predictor or uses Bayesian techniques to draw the regression coefficients β from a normal distribution centered on the estimated coefficients with a variance derived from the model uncertainty. Synthetic data are generated by comparing the modeled probabilities p with a uniform random variable, determining the binary result for each synthetic record.

- The **syn.cart** function is the default method for synthetic data generation in the package. It takes input variables y, x, xp and $\mathbf{P}$ which represent the observed response variable, observed predictors, synthetic predictors and predictor matrix respectively. This function employs the Classification and Regression Tree (CART) method to generate synthetic data.

  The process involves fitting a decision tree using the `rpart` package in R, with the flexibility to adjust parameters such as the minimum number of observations required in a tree leaf (`minbucket`) and the complexity parameter (`cp`), which influences the size of the tree. We will illustrate this in detail later in this chapter.

The function `syn.cubertnorm` synthesizes data by applying a cube root transformation to the response variable, followed by regression. The predicted values are then cubed to return to the original scale. The syn.polyreg function synthesizes categorical response variables employing Bayesian polytomous regression, fitting a multinomial logistic model and adding noise to the predicted categories. The syn.sample function generates synthetic values by randomly sampling the observed data, with optional bootstrapping to replicate the variability in the original dataset. The syn.ctree function utilizes Conditional Inference Trees to synthesize data by splitting based on conditional distributions and predicting terminal nodes for new observations. For survival data, the syn.survctree method applies Conditional Inference Trees specifically designed for time-to-event analysis, generating synthetic survival times and events. The syn.rf function synthesizes data using Random Forests, generating synthetic values from the terminal nodes of multiple decision trees. Similarly, syn.bag employs bagging (bootstrap aggregation) to fit multiple decision trees to bootstrapped samples and generate synthetic values based on aggregated predictions. Finally, syn.ipf uses Iterative Proportional Fitting (IPF) to fit a logarithmic linear model to categorical variables, adjust frequencies to match specified margins, and generate synthetic values that adhere to the distribution constraints of the original data.

Example 4.2.1. We illustrate the implementation of the package using a specific model. Assume that we have two random variables, age ($X_1 = X_{1,\text{obs}}$) and income ($X_2 = X_{2,\text{obs}}$). The sequence for this example is as follows: first, we generate the age and then generate income based on the age. Therefore, $1' = 1$ and $2' = 2$. The predictor matrix $\mathbf{P}$ is

$$\mathbf{P} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

This matrix indicates that income (X_2) is predicted using age (X_1), but age (X_1) is not predicted using any other variable. In terms of the notation introduced above, here x and x_t are identical, and it is X_1, X_2.

The basic implementation of the package is as follows.

1. Create $X_{1,\text{syn}}$ by bootstrapping $X_{1,\text{obs}}$.

2. Fit the linear model $X_{2,\text{obs}} = \beta X_{1,\text{obs}} + Z$, where Z is $\mathcal{N}(0, \sigma_Z^2)$. As a result, we obtain the estimate $\hat{\beta}$. Calculate the residuals to estimate σ_Z^2; denote it by $\hat{\sigma}_Z^2$.

3. Simulate $X_{2,\text{syn}}$ using $X_{2,\text{syn}} = \hat{\beta} X_{1,\text{syn}} + \tilde{Z}$, where $\tilde{Z}$ is centered normal with variance $\hat{\sigma}_Z^2$.

Assume that we have a dataset with 10 observations:

ID	age	income
1	22	30000
2	25	35000
3	28	40000
4	30	45000
5	35	50000
6	40	60000
7	45	70000
8	50	80000
9	55	85000
10	60	90000

In the first step, we select the initial variable: we randomly sample age_{syn} with replacement from the observed age values and obtain:

ID	age_{syn}
1	22
2	25
3	28
4	22
5	35
6	40
7	45
8	50
9	28
10	60

In the next step, we fit the linear model for the next variable $f(\text{income}_{\text{obs}} \mid \text{age}_{\text{obs}})$. The fitted model is:

$$\text{income} = 1230.77 + 1500 \cdot \text{age} + Z, \quad Z \sim N(0, \sigma^2).$$

The sum of squared residuals is 89277885.32, and the estimated σ^2 is 11159735.665. In the final step, we generate synthetic values: we draw synthetic values $\text{income}_{\text{syn}}$ from the model $f(\text{income}_{\text{syn}} \mid \text{age}_{\text{syn}})$ using normal noise with estimated σ.

The final synthetic dataset is:

Observation	age_{syn}	$\text{income}_{\text{syn}}$
1	40	60257.33
2	22	33786.17
3	30	45652.61
4	30	40345.76
5	50	75937.89
6	60	95796.99
7	30	49990.02
8	40	60031.50
9	28	47308.36
10	35	49256.03

□

We illustrate the terminology introduced before using the following example.

Example 4.2.2. Given the variables X_1, X_2, X_3, X_4 and the visiting sequence $1' = 3, 2' = 4, 3' = 2, 4' = 1$, the input predictor matrix $\mathbf{P}$ is as follows:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The rows and columns correspond to the variables X_1, X_2, X_3, X_4, and the matrix indicates which variables are used as predictors for the others.

In row 1 of the predictor matrix, we have:

$$P_{11} = 0, \quad P_{12} = 1, \quad P_{13} = 1, \quad P_{14} = 1,$$

indicating that X_2, X_3, and X_4 are predictors of X_1.

In row 2, we have:

$$P_{21} = 0, \quad P_{22} = 0, \quad P_{23} = 1, \quad P_{24} = 1,$$

indicating that X_3 and X_4 are predictors of X_2.

In row 3, we have:

$$P_{31} = 0, \quad P_{32} = 0, \quad P_{33} = 0, \quad P_{34} = 0,$$

indicating that no variables predict X_3, so the package resamples it. Notice that the variable chosen for resampling should be non-sensitive to prevent privacy disclosure.

In row 4, we have:

$$P_{41} = 0, \quad P_{42} = 0, \quad P_{43} = 1, \quad P_{44} = 0,$$

indicating that X_3 is a predictor of X_4.

Combining the predictor matrix with the visiting sequence X_3, X_4, X_2, X_1, as introduced before, we have: $1' = 3, 2' = 4, 3' = 2, 4' = 1$, which means that we generate X_3 first, X_4 second, X_2 third, and X_1 last. When synthesizing X_3, the predictor matrix indicates that only X_4 is used as a predictor (since $P_{34} = 1$). When synthesizing X_4, no predictors are used (since $P_{41} = P_{42} = P_{43} = P_{44} = 0$), which means that it is resampled. For X_2, both X_3 and X_4 are used as predictors (since $P_{23} = P_{24} = 1$). Finally, when synthesizing X_1, all variables (X_2, X_3, and X_4) are used as predictors (since $P_{12} = P_{13} = P_{14} = 1$).
Thus we have the predictor matrix as follow:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Which is same as the input predictor matrix.

In conclusion, the visiting sequence determines the order in which the variables are synthesized, while the predictor matrix defines the variables used as predictors for each step of the process. □

Example 4.2.3. We can also apply a different visiting sequence for the same input predictor matrix as shown in Example 4.2.2. For example, if we define the visiting sequence as X_4, X_3, X_1, X_2, the synthesis process changes accordingly. When synthesizing X_4, no predictors are used, even though $P_{43} = 1$ in the input predictor matrix, because X_4 is visited before X_3, meaning X_3 is resampled. For X_3, the input predictor matrix indicates that no other predictors are used, so X_3 is also resampled. When synthesizing X_1, the predictor matrix shows that X_2, X_3, and X_4 are predictors, but since only X_3 and X_4 have been visited, they are used as predictors. Finally, when synthesizing X_2, the input predictor matrix indicates that X_2 and X_3 are predictors and both have been visited. Therefore, the output predictor matrix is as follows:

$$\mathbf{P}_{\text{out}} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

In conclusion, the input predictor matrix and the output predictor matrix may be different. The input predictor matrix reflects the relationships or dependencies between variables, as defined before data generation. The output predictor matrix represents the actual model fitting during the synthesis, based on the visiting sequence and the predictors that have been visited up to that point. Therefore, while the input matrix indicates potential predictors, the output matrix shows which predictors were used in model fitting during the generation of synthetic data. □

4.3 Handling issues with data

In real applications, we have to handle various issues with the data, such as missing values, multicollinearity, or outliers. For example, to deal with linear dependence, the package proceeds as follows:

- Compute the correlation matrix $R = [R_{ij}]_{i,j=1}^{p}$ for the data and identify the pairs of variables with $|R_{ij}| \geq t$, where t is the threshold, set to 0.99999 by default.
- Adjust the predictor matrix so that the variables with high correlation are used only once.

Note that Pearson correlation primarily captures linear relationships between variables and may not effectively detect non-linear dependencies. Other types of correlation measures, such as Spearman's rank correlation or Kendall's tau, might be more appropriate. In addition, other methods, such as the variance inflation factor, may also be applied.

The package uses the `padMis.syn` function to handle missing data for continuous and `padModel.syn` function to handle missing data for discrete variable.
For continuous data the approach is as follows:

1. Check for Missing Values: Verify if the variable X_j contains missing values that are not covered by predefined rules.
2. Create Additional Variables: Create a new variable $X_j^{(0)}$ by replacing all missing values in X_j with zeros. Create another new variable $X_j^{(\mathrm{NA})}$, which is a factor variable that indicates whether the original value in X_j was missing (`Yes`) or not (`No`).
3. Update data set: Add the newly created variables $X_j^{(0)}$ and $X_j^{(\mathrm{NA})}$ as new columns in the data set.
4. Update Synthesis Methods: assign appropriate synthesis methods, distinguishing between continuous and categorical data, to newly added variables $X_j^{(0)}$ and $X_j^{(\mathrm{NA})}$.
5. Generate data: Include only newly added variables and exclude original variables with missing data when generating synthetic data.

Example 4.3.1. In this example, we demonstrate how the package handles missing data. Consider a dataset with three variables: age ($X_1 = X_{1,\mathrm{obs}}$), income ($X_2 = X_{2,\mathrm{obs}}$), and education level ($X_3 = X_{3,\mathrm{obs}}$), where some observations on education level are missing.

The original dataset is:

ID	Age (X_1)	Income (X_2)	Education (X_3)
1	25	30000	Bachelor's
2	30	40000	`NA`
3	35	50000	Master's
4	40	60000	`NA`
5	45	70000	Bachelor's
6	50	80000	High School
7	55	90000	`NA`
8	60	100000	PhD
9	65	110000	Master's
10	70	120000	`NA`

The steps to handle missing data using the `padMis.syn` function are as follows:

1. Check for Missing Values: Identify that 4 of 10 observations for 'education' are missing.

2. Create Additional Variables:

   - Create a new variable $X_3^{(0)}$ by replacing all missing values in 'education' (X_3) with zeros.
   - Create another variable $X_3^{(\text{NA})}$, which is a factor variable indicating whether the original value in 'education' was missing (`Yes`) or not (`No`).

3. Update the Dataset: Add the newly created variables $X_3^{(0)}$ and $X_3^{(\text{NA})}$ as new columns in the dataset. The updated dataset now has five columns: age (X_1), income (X_2), education (X_3), $X_3^{(0)}$, and $X_3^{(\text{NA})}$.

4. Assign Synthesis Methods:

   - Assign an appropriate synthesis method for $X_3^{(0)}$, treating it as a continuous variable or as appropriate depending on the method.
   - Assign a synthesis method for $X_3^{(\text{NA})}$, treating it as a categorical variable.

5. Generate Synthetic Data: During the synthesis process, include only the newly added variables $X_3^{(0)}$ and $X_3^{(\text{NA})}$ and exclude the original 'education' variable X_3. This approach ensures that the synthetic dataset reflects the missing data structure in the original dataset.

The updated dataset is:

ID	Age (X_1)	Income (X_2)	Education (X_3)	$X_3^{(0)}$	$X_3^{(\mathrm{NA})}$
1	25	30000	Bachelor's	1	No
2	30	40000	`NA`	0	Yes
3	35	50000	Master's	1	No
4	40	60000	`NA`	0	Yes
5	45	70000	Bachelor's	1	No
6	50	80000	High School	1	No
7	55	90000	`NA`	0	Yes
8	60	100000	PhD	1	No
9	65	110000	Master's	1	No
10	70	120000	`NA`	0	Yes

□

Discussing missing data is a broad topic and will not be covered in detail here. However, it's important to note the following specific limitations related to the handling of missing data in the current approach.

- Unspecified assumptions on missing data: The package does not explicitly specify the assumptions regarding the problem setup for the missing data. It handles missing values by replacing them with zeros and creating indicator variables, which might imply an assumption of missing completely at random (MCAR). If the data are actually missing at random (MAR) or not missing at random (NMAR), this treatment will lead to biased estimates and potentially affect the validity of the statistical analysis.

- Impact on statistical models: The use of zero-replacement and indicator variables will affect the performance of certain statistical models, particularly those sensitive to outliers or non-normal distributions.

4.4 Data utility

Once we generate the synthetic data, we need to assess whether the new data set is similar in some sense to the original data set. Recall that from the privacy perspective we do not want the original and the synthetic datasets to be similar, while the opposite holds for data utility. To assess the utility of the data generated from the statistical models, we can use two methods: first, we can directly compare the synthetic data with the original data, for example, we can apply:

- Propensity Score Analysis: This method calculates the probability that a data point is synthetic, helping to evaluate the indistinguishability of the original and synthetic data.

- Contingency tables: This method compares the frequency between groups in both observed and generated data.

- Comparison of Empirical Distributions: We compare the marginal and multivariate distributions of the synthetic and original data to evaluate the similarity between individual variables.

- Contour plots: Contour plots are used to measure the dependence structure between variables, illustrating how the generated data capture complex multidimensional relationships.

Secondly, we can also evaluate the performance of statistical models fitted to the original data to generate synthetic data. It can be evaluated using various metrics, depending on the type of data and model, for example we can apply:

- Mean squared error (MSE) for regression models. MSE is used to quantify the average squared difference between the estimated values and the actual value.

- Gini index for CART models; the Gini index measures the impurity of a node in the decision tree. Lower Gini values mean a model with better discriminative ability, which indicates a better performance.

Note that not all the tools are included in the package.

4.4.1 MSE and coefficient of determination

If the univariate distribution is generated from a statistical model, we can use the same metrics to measure the goodness of fit of the model to justify the utility of the generated data. For example, if we are using a regression model fitted by minimizing the mean squared error, we can use the mean squared error or the coefficient of determination as a utility measure:

$$\mathrm{MSE}_i := \frac{1}{n}\sum_{j=1}^{n}(X_{ij,\mathrm{obs}} - X_{ij,\mathrm{syn}})^2, \quad i = 1, \dots, p$$

and the coefficient of determination given by

$$R_i^2 = 1 - \frac{\sum_{j=1}^{n}(X_{ij,\mathrm{obs}} - X_{ij,\mathrm{syn}})^2}{\sum_{j=1}^{n}(X_{ij,\mathrm{obs}} - \bar{X}_{i,\mathrm{obs}})^2}, \quad i = 1, \dots, p,$$

where $\bar{X}_{i,\mathrm{obs}}$ is the mean of the observed values for the ith variable.

4.4.2 Comparison of marginal empirical distributions

The first measure that we can think of is the Kolmogorov-Smirnov (KS) distance between the empirical distribution of the original and the synthetic data sets. If $X_{i,\text{obs}}$ and $X_{i,\text{syn}}$ represent the ith variable for the original and synthetic data, that is, $X_{i,\text{obs}} = (X_{i1,\text{obs}}, \ldots, X_{in,\text{obs}})$ and $X_{i,\text{syn}} = (X_{i1,\text{syn}}, \ldots, X_{im,\text{syn}})$, $i = 1, \ldots, p$, then the corresponding empirical distribution functions are as follows:

$$\hat{F}_{i,\texttt{obs}}(x) = \frac{1}{n}\sum_{j=1}^{n} 1\{X_{ij,\text{obs}} \le x\} \ , \quad \hat{F}_{i,\texttt{syn}}(x) = \frac{1}{m}\sum_{j=1}^{m} 1\{X_{ij,\text{syn}} \le x\} \ .$$

The KS distance is

$$\sup_{x\in\mathbb{R}} |\hat{F}_{i,\texttt{syn}}(x) - \hat{F}_{i,\texttt{obs}}(x)| \ .$$

This approach has been used in many papers on data privacy, for example in [37]. However, note that we can use the KS distance only as a visual "test of goodness of fit". The classical results on the asymptotic behavior of the KS statistics are not applicable here since the observed and the synthetic data are dependent.

Example 4.4.1. In this example, we illustrate that using the KS-statistics in the context of synthetic data may not be appropriate due to the dependent structure between the generated data and the original data.

Suppose that we have the following two random variables, x_1 and x_2. Let x_1 follow a standard normal distribution $N(0, 1)$, and x_2 is generated from x_1 using the linear relationship:

$$x_2 = \beta_0 + \beta_1 x_1 + \epsilon,$$

where $\beta_0 = 0.2$, $\beta_1 = 0.8$, and ϵ is normally distributed with zero mean and a variance of 0.36, $\epsilon \sim N(0, 0.36)$.

The covariance matrix Σ for $[x_1, x_2]$ is calculated as follows:

- Variance of x_1:

$$\sigma^2_{x_1} = 1$$

- Variance of x_2 :

$$\sigma^2_{x_2} = \beta_1^2 \sigma^2_{x_1} + \sigma^2_{\epsilon} = 0.8^2 \cdot 1 + 0.36 = 1.0$$

- Covariance between x_1 and x_2 :

$$\sigma_{x_1,x_2} = \beta_1 \sigma^2_{x_1} = 0.8 \cdot 1 = 0.8$$

The covariance matrix Σ is then:

$$\Sigma = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix} .$$

The null hypothesis is: *there is no difference between the distribution functions of the original and the synthetic data*:

$$H_0 : F(x) = G(x) \text{ for all } x.$$

The alternative hypothesis is that there is a difference at least at one point:

$$H_a : F(x) \neq G(x) \text{ for at least one } x.$$

The null distribution of the KS statistic between two datasets is based on the maximum difference in their empirical cumulative distribution functions (ECDFs):

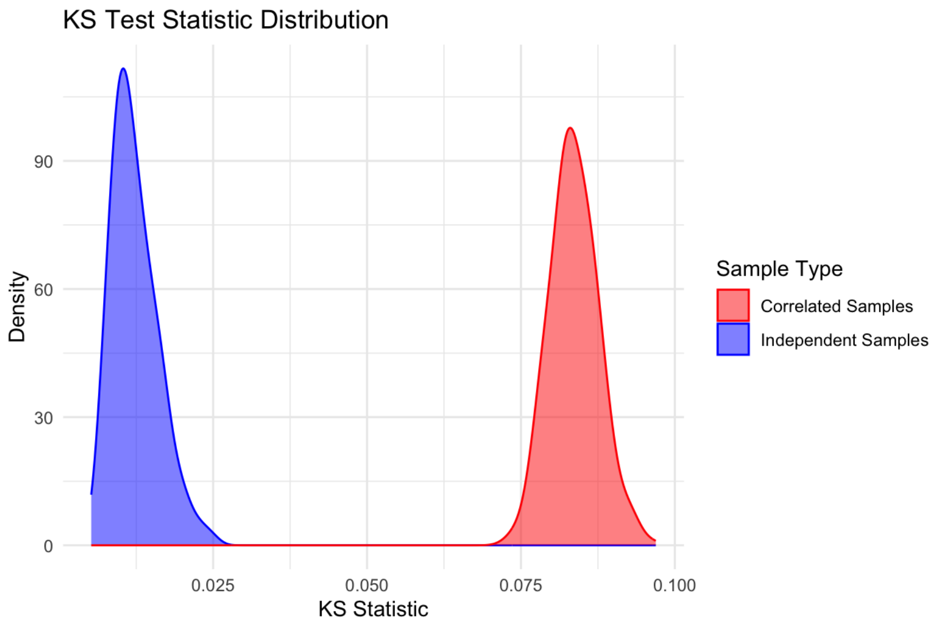
$$K = \sup_x |\hat{F}_n(x) - \hat{G}_m(x)|$$

where $\hat{F}_n$ and $\hat{G}_m$ are the ECDFs of the original and synthetic data, respectively.

Under the assumption of independence, the distribution of K under the null hypothesis follows:

$$K(x) = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1}e^{-2k^2x^2}.$$

However, in our case, x_2 is directly influenced by x_1, creating a dependent structure. This dependence changes the behavior of the ECDFs, leading to a different distribution of the KS statistic, as shown in the simulation results below.



The Kolmogorov-Smirnov test assumes independent samples, so applying it to dependent data can lead to misleading results.

4.4.3 Contour Plots

An alternative measure of utility could be based on the comparison of contour plots for dependence structures. Contour plots are a graphical representation used to illustrate the joint distribution of two continuous random variables X and Y. Mathematically, the contour plot is constructed by plotting the level curves of the joint PDF:

$$f_{X,Y}(x, y) = c,$$

where c is a constant representing a specific probability density value. Each contour line corresponds to a different value of c.

To calculate contour plots for both original and generated data, we use Kernel Density Estimation (KDE) to smooth them. KDE is a nonparametric method for estimating the probability density function of a random variable. For a pair of continuous random variables X and Y, the KDE of the joint PDF is given by:

$$\hat{f}_{X,Y}(x, y) = \frac{1}{nh_X h_Y} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h_X}\right) K\left(\frac{y - Y_i}{h_Y}\right),$$

where n is the number of data points, h_X and h_Y are the bandwidth parameters for X and Y respectively, and $K(\cdot)$ is the kernel function, typically chosen to be the Gaussian kernel:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}.$$

4.4.4 Propensity score

We first introduce a propensity score-based measure. The propensity score is commonly used in the context of causal inference, where it represents the probability of receiving a treatment given a set of observed covariates. Similarly, in missing data scenarios, the propensity score estimates the probability that a particular individual's data is missing [24].

In the context of synthetic data generation, the propensity score is a utility measure to assess the similarity between the original and synthetic data. This is conceptually similar to the role of the discriminator in generative adversarial networks (GANs), which differentiate between original and synthetic data. However, unlike GANs, where the discriminator is part of the model training process, the propensity score here is used after data generation to evaluate the quality of synthetic data.

The propensity score is the conditional probability that a data point is synthetic given the observed and synthetic datasets. The process involves first combining the synthetic and original datasets and then adding an indicator variable to each data point to denote

whether it is synthetic or not. This indicator variable is then used as the response to fit a model, such as logistic regression (logit) or classification trees (CART), with all other observed variables predicting the indicator. The fitted model provides predicted probabilities for each data point of being synthetic or original, which are the propensity scores. These scores represent the likelihood that each data point is synthetic on the basis of its characteristics, helping to measure the similarity between synthetic and observed data. The idea is that if the synthetic data is similar to observed data, it should be difficult to distinguish between them by propensity score. Several papers discussed the use of propensity score and its statistical properties, including test statistics, [30].
Here we discuss the propensity score used in the package. In what follows:

- n is the number of observed units;
- m is the number of synthetic units;
- $N = n + m$ is the total number of observations;
- $c = \frac{m}{N}$ is the proportion of synthetic units;
- t is an indicator variable that distinguishes between synthetic and observed data. It is added to the combined dataset to indicate which records are synthetic ($t = 1$) and which are observed ($t = 0$).
  - Logistic regression (Logit): Uses the glm function to fit a logistic regression model $t = f(x)$. The function $f(x)$ represents the logistic regression model, where X is the vector of features and $f(x)$ gives the probability that an observation is synthetic.
  - Classification and Regression Trees (CART): Uses either the rpart or ctree function to fit a classification tree $t = f(x)$. The function $f(x)$ represents the classification tree, where X is the vector of characteristics, and $f(x)$ gives the probability that an observation is synthetic.
- $\hat{p}_j$ is the propensity score for the jth observation, which is the predicted probability using model $f(x)$ introduced above. If propensity scores are close to c, then the assignment of these scores is essentially random. Then it indicates that it is difficult to distinguish between synthetic and original data. In turn, it means good data utility.

The package provides the following measure based on the propensity score $\hat{p}(x_j)$.

- pMSE (Propensity Mean Squared Error)
  - pMSE measures the mean squared error of propensity scores, lower pMSE indicates better similarity between synthetic and observed data. The package does not use a test statistic that returns a p-value for pMSE.

- Formula:
$$\text{pMSE} = \frac{1}{N}\sum_{j=1}^{N}(\hat{p}_j - c)^2.$$
- A lower pMSE value indicates that the propensity scores are very close to c, suggesting that it is difficult to distinguish between synthetic and original data.

- SPECKS (Kolmogorov-Smirnov Statistic)
  - The Kolmogorov-Smirnov (KS) statistic measures the maximum difference between the empirical cumulative distribution functions of the propensity scores for observed and synthetic data. It is a statistics designed to test if two independent samples come from the same distribution. The package uses the KS test, which returns a p-value.
  - Formula: Denote $\hat{p}_{(x_j,\text{obs})}$ and $\hat{p}_{(x_j,\text{syn})}$ the propensity score for data point from observed and synthetic dataset respectively, $F_{\text{obs}}(x)$ and $F_{\text{syn}}(x)$ are the empirical cumulative distribution functions of the propensity scores for the observed and synthetic data, where
  $$F_{\text{obs}}(x) = \frac{1}{n}\sum_j 1\{\hat{p}_{(x_j,\text{obs})} \le x\} \text{ and } F_{\text{syn}}(x) = \frac{1}{m}\sum_j 1\{\hat{p}_{(x_j,\text{syn})} \le x\},$$
  for $x \in (0,1)$. We have
  $$\text{SPECKS} = \sup_x |F_{\text{obs}}(x) - F_{\text{syn}}(x)|.$$
  - p-Value: The package uses the KS test, which returns a p-value:
  ```
  KSt<-ks.test(score[df.prop$t == 1], score[df.prop$t == 0])
  SPECKS<-KSt$statistic
  ```
  As mentioned above, the classical KS test is for independent random samples. Here the propensity scores are for synthetic and observed data, hence the produced p-values are <u>likely incorrect</u>.

- Wilcoxon Statistic
  - The Wilcoxon rank-sum test is a nonparametric equivalent of the paired t-test. It makes no assumption about the distributions of the original population themselves, but it does assume that the distributions of the differences are at least symmetric.

- Formula:

$$\text{U} = \sum_j \text{Rank}(\hat{p}_j^{\text{obs}}) - \frac{n(n+1)}{2}.$$

  where $\hat{p}_j^{\text{obs}}$ are the propensity scores for the observed data, and n is the number of observed units.

- p-Value: The package uses the Wilcoxon test, which returns a p-value:

```
U<-wilcox.test(score[df.prop$t == 1], score[df.prop$t == 0])$statistic
```

  The package uses the Wilcoxon rank-sum test which is designed for independent samples. As discussed before, since the propensity scores come from both synthetic and observed data, the resulting p-values may not be accurate.

Example 4.4.2. Based on Example 4.2.1 now assume that we have 3 random variables: age ($X_{1,\text{obs}}$), income ($X_{2,\text{obs}}$), and sex ($X_{3,\text{obs}}$). The visit sequence for this example involves first generating age, then generating income based on age, and finally generating gender based on age and income. The basic implementation of the package is as follows.

1. Create $X_{1,\text{syn}}$ by bootstrapping $X_{1,\text{obs}}$.

2. Fit the linear model $X_{2,\text{obs}} = \beta X_{1,\text{obs}} + Z$. As a result, obtain the estimate $\hat{\beta}$.

3. Simulate $X_{2,\text{syn}}$ using $X_{2,\text{syn}} = \hat{\beta} X_{1,\text{syn}} + \tilde{Z}$, where $\tilde{Z}$ is another random noise.

4. Centering the data involves subtracting the mean of each predictor variable from the observed and synthetic values. The centered predictor variables are defined as follows:

$$\begin{aligned}
X_{1,\text{obs, centered}} &= X_{1,\text{obs}} - \bar{X}_{1,\text{obs}},\\
X_{2,\text{obs, centered}} &= X_{2,\text{obs}} - \bar{X}_{2,\text{obs}},\\
X_{1,\text{syn, centered}} &= X_{1,\text{syn}} - \bar{X}_{1,\text{syn}},\\
X_{2,\text{syn, centered}} &= X_{2,\text{syn}} - \bar{X}_{2,\text{syn}}.
\end{aligned}$$

5. Fit a logistic model $X_{3,\text{obs}} = f(\beta_1 X_{1,\text{obs,centered}} + \beta_2 X_{2,\text{obs,centered}})$. As a result, obtain the estimates $\hat{\beta}_1$ and $\hat{\beta}_2$.

6. Generate $X_{3,\text{syn}}$ using the logistic model $X_{3,\text{syn}} = f(\tilde{\beta}_1 X_{1,\text{syn,centered}} + \tilde{\beta}_2 X_{2,\text{syn,centered}})$, where $\tilde{\beta}_1$ and $\tilde{\beta}_2$ are drawn from the distribution $\mathcal{N}(\hat{\beta}_1, V(\hat{\beta}_1))$ and $\mathcal{N}(\hat{\beta}_2, V(\hat{\beta}_2))$, respectively, using a Bayesian method.

We already have the model for income as calculated from the previous example:

$$\text{income}_{\text{centered}} = 1230.77 + 1500 \cdot \text{age}_{\text{centered}} + Z, \quad Z \sim N(0, \sigma^2).$$

ID	**Age**	**Income**	**Gender**
1	22	30000	M
2	25	35000	F
3	28	40000	M
4	30	45000	F
5	35	50000	M
6	40	60000	F
7	45	70000	M
8	50	80000	F
9	55	85000	M
10	60	90000	F

Table 4.1: Demographic Data of Individuals

Now we center the data and fit a logistic regression model

$$f(\text{gender}_{\text{obs}} \mid \text{age}_{\text{obs,centered}}, \text{income}_{\text{obs,centered}}).$$

The model is:

$$\log\left(\frac{\mathbb{P}(\text{gender}=1 \mid X_{1,\text{obs, centered}}, X_{2,\text{obs, centered}})}{\mathbb{P}(\text{gender}=0 \mid X_{1,\text{obs, centered}}, X_{2,\text{obs, centered}})}\right) = \beta_0 + \beta_1 X_{1,\text{obs, centered}} + \beta_2 X_{2,\text{obs, centered}},$$

and fitted model is

$$\log\left(\frac{\mathbb{P}(\text{gender}=1 \mid \text{age}_{\text{centered}}, \text{income}_{\text{centered}})}{\mathbb{P}(\text{gender}=0 \mid \text{age}_{\text{centered}}, \text{income}_{\text{centered}})}\right) = -1.5 + 0.3 \cdot \text{age}_{\text{centered}} + 0.5 \cdot \text{income}_{\text{centered}}.$$

The variances for the coefficients are calculated as follows:

$$\text{Var}(\hat{\beta}_0) = 0.749956,$$
$$\text{Var}(\hat{\beta}_1) = 0.040000,$$
$$\text{Var}(\hat{\beta}_2) = 0.090000.$$

The coefficients are assumed to follow normal distributions as follows:

$$\hat{\beta}_0 \sim \mathcal{N}(-1.5, 0.749956),$$
$$\hat{\beta}_1 \sim \mathcal{N}(0.3, 0.040000),$$
$$\hat{\beta}_2 \sim \mathcal{N}(0.5, 0.090000).$$

Using these distributions, we draw one sample for each β:

$$\text{Sample for } \hat{\beta}_0 : -1.583315,$$
$$\text{Sample for } \hat{\beta}_1 : 0.4983624,$$
$$\text{Sample for } \hat{\beta}_2 : 0.1630316.$$

Table 4.2: Synthetic Dataset

Observation	Age Syn	Income Syn	Generated Gender
1	40	60257.33	F
2	22	33786.17	M
3	30	45652.61	M
4	30	40345.76	M
5	50	75937.89	F
6	60	95796.99	F
7	30	49990.02	M
8	40	60031.50	F
9	28	47308.36	M
10	35	49256.03	M

Table 4.3: Combined Original and Synthetic Dataset with Indicator

ID	Age	Income	Gender	Indicator (t)
1	40	60257.33	F	1
2	22	33786.17	M	1
3	30	45652.61	M	1
4	30	40345.76	M	1
5	50	75937.89	F	1
6	60	95796.99	F	1
7	30	49990.02	M	1
8	40	60031.50	F	1
9	28	47308.36	M	1
10	35	49256.03	M	1
11	22	30000	M	0
12	25	35000	F	0
13	28	40000	M	0
14	30	45000	F	0
15	35	50000	M	0
16	40	60000	F	0
17	45	70000	M	0
18	50	80000	F	0
19	55	85000	M	0
20	60	90000	F	0

The generated data set is then

Then we fit a logistic regression model for t. The logistic regression model for the propensity score is:

$$\log\left(\frac{\mathbb{P}(t=1 \mid \text{age, income, gender})}{\mathbb{P}(t=0 \mid \text{age, income, gender})}\right) = \gamma_0 + \gamma_1 \cdot \text{age} + \gamma_2 \cdot \text{income} + \gamma_3 \cdot \text{gender},$$

where

$$\hat{\gamma}_0 = -0.8473, \quad \hat{\gamma}_1 = 0.0147, \quad \hat{\gamma}_2 = 0.0001, \quad \hat{\gamma}_3 = 0.6981 \quad .$$

The calculated propensity scores are:

Table 4.4: Dataset with Propensity Scores

ID	Age	Income	Gender	Indicator	Propensity Score
1	40	60257.33	F	1	0.4384
2	22	33786.17	M	1	0.6838
3	30	45652.61	M	1	0.6064
4	30	40345.76	M	1	0.3405
5	50	75937.89	F	1	0.3782
6	60	95796.99	F	1	0.5286
7	30	49990.02	M	1	0.7902
8	40	60031.50	F	1	0.4270
9	28	47308.36	M	1	0.8130
10	35	49256.03	M	1	0.3623
11	22	30000.00	M	0	0.4978
12	25	35000.00	F	0	0.4424
13	28	40000.00	M	0	0.4909
14	30	45000.00	F	0	0.5222
15	35	50000.00	M	0	0.3984
16	40	60000.00	F	0	0.4254
17	45	70000.00	M	0	0.5569
18	50	80000.00	F	0	0.5842
19	55	85000.00	M	0	0.4598
20	60	90000.00	F	0	0.2535

Given $N = 20$ and $c = \frac{10}{20} = 0.5$, the Propensity Mean Squared Error (pMSE) is calculated as:

$$\text{pMSE} = \frac{1}{20}\sum_{j=1}^{20}(\hat{p}_j - 0.5)^2 = 0.0194. \tag{4.1}$$

Example 4.4.3. In this example we illustrate potential issues with propensity scores. We start with the same original data set as above. The synhetic data set is the same

Table 4.5: Combined Dataset with Indicator

ID	**Age**	**Income**	**Gender**	**Indicator** (t)
1	22	30000	M	1
2	25	35000	F	1
3	28	40000	M	1
4	30	45000	F	1
5	35	50000	M	1
6	40	60000	F	1
7	45	70000	M	1
8	50	80000	F	1
9	55	85000	M	1
10	60	90000	F	0
11	22	30000	M	0
12	25	35000	F	0
13	28	40000	M	0
14	30	45000	F	0
15	35	50000	M	0
16	40	60000	F	0
17	45	70000	M	0
18	50	80000	F	0
19	55	85000	M	0
20	60	90000	F	0

as the original data set. We fit another logistic regression model for t. The logistic regression model for the propensity score is:

$$\log\left(\frac{\mathbb{P}(t=1 \mid \text{age, income, gender})}{\mathbb{P}(t=0 \mid \text{age, income, gender})}\right) = \gamma\prime_0 + \gamma\prime_1 \cdot \text{age} + \gamma\prime_2 \cdot \text{income} + \gamma\prime_3 \cdot \text{gender},$$

where

$$\hat{\gamma}\prime_0 = 0.219, \quad \hat{\gamma}\prime_1 = -0.243, \quad \hat{\gamma}\prime_2 = 0.00015, \quad \hat{\gamma}\prime_3 = 0.312 \quad .$$

The calculated propensity scores are: Given $N = 20$ and $c = \frac{10}{20} = 0.5$, the Propensity

Table 4.6: Dataset with Propensity Scores

ID	Age	Income	Gender	Indicator	Propensity Score
1	22	30000	M	1	0.5906
2	25	35000	F	1	0.4523
3	28	40000	M	1	0.5546
4	30	45000	F	1	0.5346
5	35	50000	M	1	0.4002
6	40	60000	F	1	0.3638
7	45	70000	M	1	0.5634
8	50	80000	F	1	0.5252
9	55	85000	M	1	0.3911
10	60	90000	F	0	0.1241
11	22	30000	M	0	0.5906
12	25	35000	F	0	0.4523
13	28	40000	M	0	0.5546
14	30	45000	F	0	0.5346
15	35	50000	M	0	0.4002
16	40	60000	F	0	0.3638
17	45	70000	M	0	0.5634
18	50	80000	F	0	0.5252
19	55	85000	M	0	0.3911
20	60	90000	F	0	0.1241

Mean Squared Error (pMSE) is calculated as:

$$\text{pMSE} = \frac{1}{20}\sum_{j=1}^{20}(\hat{p}_j - 0.5)^2 = 0.02009.$$

Now, we compare this expression with (4.1). Notice that the pMSE values for two different synthetic datasets are similar, even though the synthetic data sets are produced in a completely different way. Furthermore, if we consider another situation where the synthetic and original are the same, we will get a different value of pMSE. It suggests that pMSE may not reflect the true utility of the data. One possible reason for this could be that the small size of the dataset leads to poor fitting of the propensity score model. $\square$

Example 4.4.4. We now demonstrate the influence of different synthetic datasets on the calculation of propensity scores mean square error (pMSE). We consider two types of datasets generated from a multivariate normal distribution with a covariance matrix

$$\Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix},$$

but with different means, where each dataset contains 100 data points:

- X_{obs} contains variables X_1, X_2 with mean
$$\mu = (0, 0),$$
representing the observed data.
- X_{syn} also contains variables X_1, X_2, where the mean
$$\mu = (a, a)$$
varies as a changes from 0 to 20, representing synthetic data.

The idea is that as a gets bigger, the synthetic data set will be more different from the original data, hence more private. For each value of a, the two data sets are combined and an indicator variable (0 for observed data, 1 for synthetic data) is added. A logistic regression model is fitted to this combined dataset, using the formula:

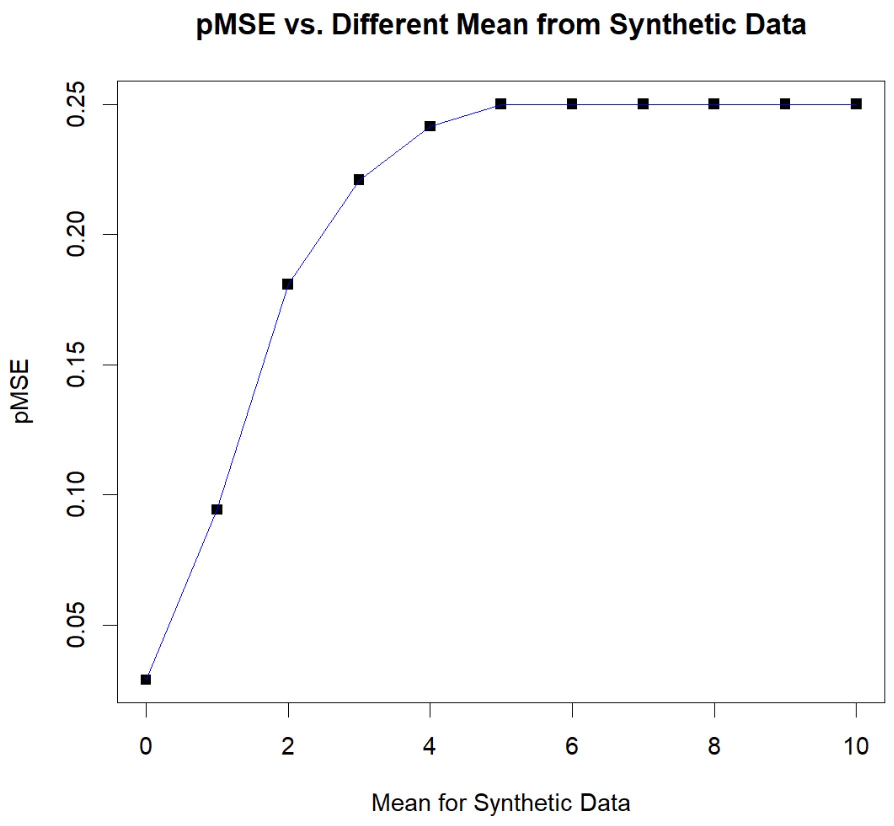
$$\operatorname{logit}(\mathbb{P}(\text{Indicator} = 1)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2,$$

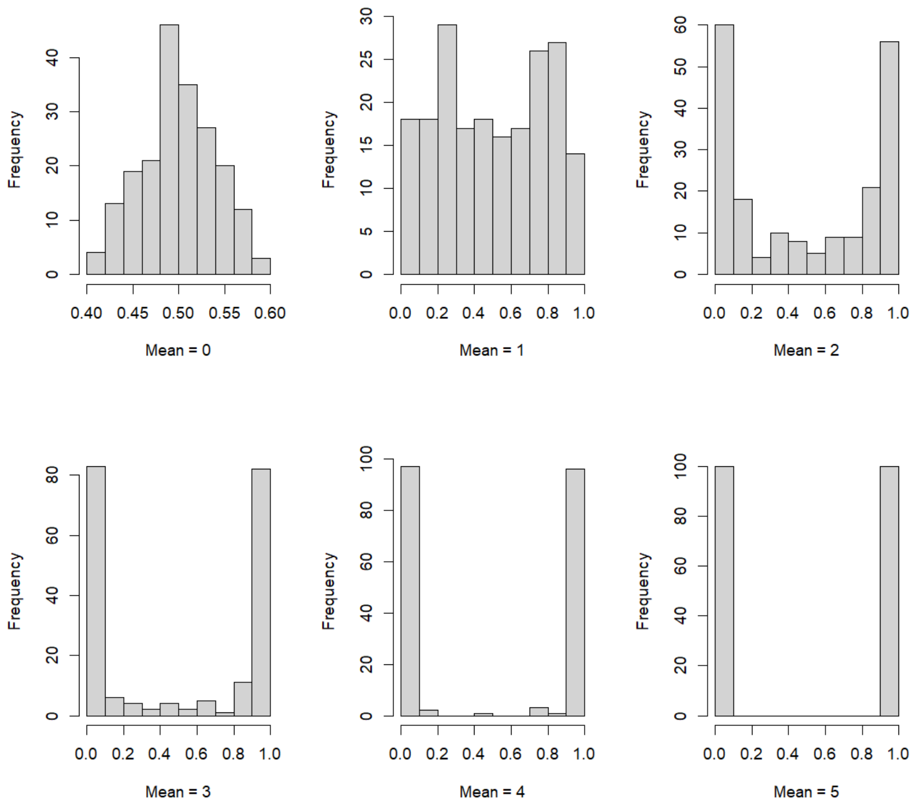
and the propensity score, $\hat{p}_j$, is computed for each observation.

We then calculate the Propensity Mean Squared Error (pMSE) for each dataset by averaging the squared differences between the propensity scores and 0.5:

$$\text{pMSE}(a) = \frac{1}{200} \sum_{j=1}^{200} (\hat{p}_j - 0.5)^2.$$

The resulting pMSE values illustrate how the similarity or dissimilarity between the observed and synthetic data affects the propensity score's accuracy and the model's utility. The pMSE values for each a are plotted against a to visually show this relationship.


pMSE vs. Different Mean from Synthetic Data
pMSE
0.05
0.10
0.15
0.20
0.25
0
2
4
6
8
10
Mean for Synthetic Data



The experiment shows that as a increases, the mean squared error (MSE) of the propensity scores rises from 0, reflecting greater detectability of synthetic data. In the worst-case scenario, the MSE converges to 0.25, since the propensity score would randomly assign data according to the proportion of synthetic data, leading to an error of $0.5 \times 0.5 = 0.25$. $\square$

Proper test statistics for the propensity score The package does not provide the proper test statistics for the propensity score. [30] proposed a test statistics using the propensity score when using logistic regression to fit the score. We have

- Null hypothesis: original data and synthetic data cannot be distinguished by logistic regression model.
- Alternative hypothesis: original data and synthetic data can be distinguished by a logistic regression model.

The test statistics used is the propensity score mean squared error:

$$\text{pMSE} = \frac{1}{N}\sum_{j=1}^{N}(\hat{p}_j - c)^2.$$

Under null hypothesis its follows a chi-squared distribution with $(k-1)$ degrees of freedom, where k is the number of predictors used in the logistic regression fitting the propensity score, the means and variance are given as follows:

$$\mathbb{E}[\text{pMSE}] = (k-1)\left(\frac{n_1}{N}\right)^2\left(\frac{n_2}{N}\right)\frac{1}{N} = (k-1)(1-c)^2 c\frac{1}{N},$$

$$\text{StDev}(\text{pMSE}) = \sqrt{2(k-1)\left(\frac{n_1}{N}\right)^2\left(\frac{n_2}{N}\right)\frac{1}{N}} = \sqrt{2(k-1)(1-c)^2 c\frac{1}{N}}.$$

4.4.5 Contingency Table

A contingency table is a type of table that displays the frequency distribution of multivariate observations. Each cell in the table represents the count of observations for a specific combination of categories. If the variables are continuous, they are grouped into categories before building the table; by default in the package, they are grouped into five groups. In what follows,

- Let C be the set of all possible combinations of categorical and grouped continuous variables.
- For each $c \in C$:
  - $td(c)$ is the count of data points in c from the original data. That is

$$td(c) = \sum_{j=1}^{n} \mathbf{1}((X_{1j,\text{obs}}, \ldots, X_{pj,\text{obs}}) \in c).$$

  - $ts(c)$ is the count of data points in c from the synthetic data. That is

$$ts(c) = \sum_{j=1}^{m} \mathbf{1}((X_{1j,\text{syn}}, \ldots, X_{pj,\text{syn}}) \in c).$$

The package provides the following measures based on the contingency table.

- VW (Variance-Weighted Measure)
  - Variance-Weighted Measure assesses the difference between observed and synthetic counts. It calculates the sum of squared differences between observed and expected counts, weighted by the expected counts.
  - Formula:

$$\text{VW} = \sum_{c\in C}\left(\frac{(td(c)-ts(c))^2}{ts(c)}\right).$$

- JSD (Jensen-Shannon Divergence)

- JSD is a symmetrized version of the Kullback-Leibler divergence that measures the similarity between two probability distributions. A JSD of 0 means that the two distributions are identical, while a JSD of 1 indicates maximum divergence.
- Formula:
$$\frac{1}{2}\sum_{c\in C} p(c)\log_2\left(\frac{2p(c)}{p(c)+q(c)}\right)+\frac{1}{2}\sum_{c\in C} q(c)\log_2\left(\frac{2q(c)}{p(c)+q(c)}\right),$$
where $p(c)=\frac{td(c)}{\sum_{c\in C} td(c)}$ and $q(c)=\frac{ts(c)}{\sum_{c\in C} ts(c)}$.

- SPECKS (Kolmogorov-Smirnov Statistic)
  - This statistic measures the maximum difference between the cumulative distributions of observed and synthetic data. It assesses how well the synthetic data's distribution matches the real data's distribution.
  - Formula:
$$\max_{c\in C}\left|\frac{\sum_{i\in c} td(i)}{\sum_{i\in C} td(i)}-\frac{\sum_{i\in c} ts(i)}{\sum_{i\in C} ts(i)}\right|.$$
  - p-value: Provided by `ks.test`. Again, the p-value seems to be incorrect given the dependence between the observed and synthetic data.

The closer the value to zero indicates better utility.

Example 4.4.5. We use the same data from Example 4.4.2. Given the continuous nature of the data, we categorize it into two groups (unlike the package does): Group 1 includes income with values that are less than or equal to \$60,000, and Group 2 includes values that are greater than \$60,000.

Variance-Weighted Measure (VW):
$$VW=\frac{(6-5)^2}{5}+\frac{(4-5)^2}{5}=0.4.$$

Jensen-Shannon Divergence (JSD): Using $p(c)=\frac{td(c)}{10}$ and $q(c)=\frac{ts(c)}{10}$,
$$\frac{1}{2}\left[\frac{6}{10}\log_2\left(\frac{12}{11}\right)+\frac{5}{10}\log_2\left(\frac{10}{11}\right)\right]+\frac{1}{2}\left[\frac{4}{10}\log_2\left(\frac{8}{9}\right)+\frac{5}{10}\log_2\left(\frac{10}{9}\right)\right].$$

SPECKS (Kolmogorov-Smirnov Statistic):
$$\max\left(\left|\frac{6}{10}-\frac{5}{10}\right|,\left|\frac{10}{10}-\frac{10}{10}\right|\right)=0.1.$$

A measure closer to zero indicates better utility; however, these numbers lack precise interpretations as they are just measures of distance. No proper test statistics are provided. □

Table 4.7: Dataset with Grouping

ID	**Age**	**Income**	**Gender**	**Group** (c)
1	40	60257.33	F	2
2	22	33786.17	M	1
3	30	45652.61	M	1
4	30	40345.76	M	1
5	50	75937.89	F	2
6	60	95796.99	F	2
7	30	49990.02	M	1
8	40	60031.50	F	2
9	28	47308.36	M	1
10	35	49256.03	M	2
11	22	30000	M	1
12	25	35000	F	1
13	28	40000	M	1
14	30	45000	F	1
15	35	50000	M	1
16	40	60000	F	1
17	45	70000	M	2
18	50	80000	F	2
19	55	85000	M	2
20	60	90000	F	2

Table 4.8: Synthetic and Observed Counts for Groups

Group (c)	**Synthetic Count** ($ts(c)$)	**Observed Count** ($td(c)$)
1	5	6
2	5	4

Table 4.9: Synthetic and Observed Counts for Groups

Group	**Synthetic Count** ($ts(c)$)	**Observed Count** ($td(c)$)
1	6	6
2	4	4

Example 4.4.6. Continuing from the previous example, now suppose original data is released as synthetic data, we have the following:

Variance-Weighted Measure (VW):

$$VW = \frac{(6-6)^2}{6} + \frac{(4-4)^2}{4} = 0.$$

Jensen-Shannon Divergence (JSD):

$$JSD = \frac{1}{2}\left[\frac{6}{10}\log_2(1) + \frac{4}{10}\log_2(1)\right] + \frac{1}{2}\left[\frac{6}{10}\log_2(1) + \frac{4}{10}\log_2(1)\right] = 0.$$

SPECKS (Kolmogorov-Smirnov Statistic):

$$\max\left(\left|\frac{6}{10} - \frac{6}{10}\right|, \left|\frac{10}{10} - \frac{10}{10}\right|\right) = 0.$$

Note that all the measures are equal to zero, the synthetic data and the original data are identical. Thus, a measure closer to zero indicates better utility. □

Example 4.4.7. Given the same dataset as before, we now categorize the income data into three different groups to see how this affects the calculation of our statistical measures (VW, JSD, and SPECKS).

- Variance-Weighted Measure (VW):

$$VW = \frac{(4-8)^2}{8} + \frac{(10-10)^2}{10} + \frac{(4-2)^2}{2} = 2 + 0 + 4 = 6.$$

- Jensen-Shannon Divergence (JSD):

$$\begin{aligned} &\frac{1}{2}\left[\frac{4}{18}\log_2\left(\frac{2\times\frac{4}{18}}{\frac{4}{18}+\frac{8}{18}}\right) + \frac{8}{18}\log_2\left(\frac{2\times\frac{8}{18}}{\frac{4}{18}+\frac{8}{18}}\right)\right] + \dots \\ &+ \frac{1}{2}\left[\frac{10}{18}\log_2\left(\frac{2\times\frac{10}{18}}{\frac{10}{18}+\frac{10}{18}}\right) + \frac{10}{18}\log_2\left(\frac{2\times\frac{10}{18}}{\frac{10}{18}+\frac{10}{18}}\right)\right] + \dots \\ &+ \frac{1}{2}\left[\frac{4}{18}\log_2\left(\frac{2\times\frac{4}{18}}{\frac{4}{18}+\frac{2}{18}}\right) + \frac{2}{18}\log_2\left(\frac{2\times\frac{2}{18}}{\frac{4}{18}+\frac{2}{18}}\right)\right]. \end{aligned}$$

- SPECKS (Kolmogorov-Smirnov Statistic):

$$\max\left(\left|\frac{4}{18} - \frac{8}{18}\right|, \left|\frac{10}{18} - \frac{10}{18}\right|, \left|\frac{4}{18} - \frac{2}{18}\right|\right) = \max\left(\frac{4}{18}, 0, \frac{2}{18}\right) = \frac{4}{18}.$$

These calculations illustrate how distance measures fluctuate based on the method used to group data. Specifically, when a single group is selected, the distance measure will always be zero, regardless of how the data are generated, indicating perfect data utility. When multiple groups are used, even a single difference will lead to a large distance, thus underestimating the utility of the data. □

Table 4.10: Income Grouping

ID	Income	Group by $60,000	New Grouping
1	60257.33	2	2
2	33786.17	1	1
3	45652.61	1	2
4	40345.76	1	2
5	75937.89	2	2
6	95796.99	2	3
7	49990.02	1	2
8	60031.50	2	2
9	47308.36	1	2
10	49256.03	2	2
11	30000	1	1
12	35000	1	1
13	40000	1	1
14	45000	1	2
15	50000	1	2
16	60000	1	2
17	70000	2	2
18	80000	2	2
19	85000	2	3
20	90000	2	3

4.4.6 Permutation test

A permutation test is a non-parametric method used to test null hypotheses of equality between two distributions by simulating a distribution of the test statistic. The process can be described as follows:

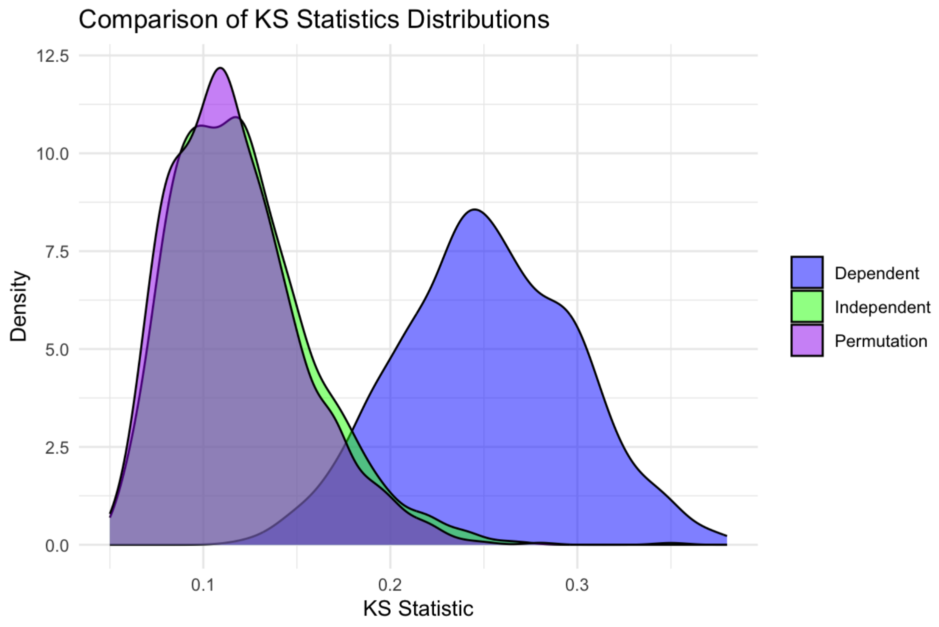
- Choose an appropriate test statistic.
- The test statistic is then computed for the original labeling of the observations.
- Then permute the labels of the data and recompute the test statistic for each permutation.
- The above step is repeated many times to generate a distribution of the test statistic under the null hypothesis. The null hypothesis in the permutation test is usually stated because there is no effect or no difference between the groups (labels) being compared.
- Finally, the observed test statistic is compared to this permutation distribution to decide whether to accept or reject the null hypothesis.

If the observed statistic is significantly different from the permutation distribution, the null hypothesis is rejected, indicating a statistically significant effect. Both propensity-based and contingency table-based scores can be evaluated using the permutation test with the code `resamp.method == "perm"`, the package set original number of permutation as 30.

Example 4.4.8. In this experiment we generate data from a multivariate normal distribution with mean $\mu = (50, 55)$ and covariance matrix

$$\Sigma = \begin{pmatrix} 100 & 50 \\ 50 & 100 \end{pmatrix}.$$

We apply a permutation test by randomly permuting the group labels of the dataset. For each permutation, we calculate the KS statistic, K, based on the permuted data. The distribution of the KS statistics, K, is shown below. Additionally, we repeat this process to calculate K under the assumption of independence, as well as under the true dependency structure of the data (as described in Example 4.4.1). This allows us to compare the distribution of K between these different assumptions.



We notice that the permutation test is still only valid under the assumption of independence, and thus may not be appropriate for testing contingency tables as it was used in the package. However, it can be applied to the propensity score when fitting the CART model, as shown in [30]. $\square$

4.4.7 Gini index

Gini index is a special data utility measure that applies to data with a tree-like structure. Assume we have T nodes and K_t classes in node t. A class is a category or a label in which instances are classified, and a node is a point in the classification tree where data are split based on certain features. In a classification tree, the Gini index is the measure of impurity in a node, reflecting the probability that a randomly chosen instance would be incorrectly classified if it were randomly labeled according to the proportions of classes in the node. Let $p_{k,t}$ denote the proportion of instances of class k in node t, this proportion also represents the probability of randomly choosing an instance of class k in node t. Therefore, $p_{k,t}^2$ represents the probability that a randomly chosen instance is of class k and is also correctly classified as class k. The probability that a randomly chosen instance is assigned to incorrect classes (i.e., misclassified) is the complement of the probability that they are of the same class. This relationship is captured by the Gini index as follows:

$$\text{Gini}(t) = 1 - \sum_{k=1}^{K} p_{k,t}^2.$$

Purity in decision trees refers to how homogeneous or uniform the instances in a node are regarding their class labels. A node is considered pure if all instances belong to a

single class, ex. $\text{Gini}(t) = 0$.
A lower gini index indicates a better classification because it indicates a purer node. Conversely, a higher Gini index indicates a more mixed node, with a greater diversity of classes present, leading to a higher likelihood of misclassification.

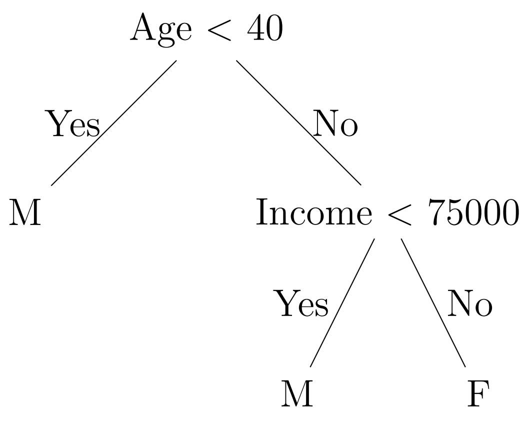
Example 4.4.9. Assume we have the following data:

Table 4.11: Demographic Data

ID	**Age**	**Income**	**Gender**
1	22	30000	M
2	25	35000	F
3	28	40000	M
4	30	40000	F
5	40	50000	M
6	40	60000	M
7	45	80000	M
8	50	80000	F
9	55	85000	F
10	60	90000	F

We have the model as follows:

- Goal: Predict the gender (M/F) based on age and income.
- Tree Model:



- Root Node (all data):
  - Total instances: 10
  - Males (M): 5, Females (F): 5
  - $p_{M,\text{root}} = \frac{5}{10} = 0.5, \quad p_{F,\text{root}} = \frac{5}{10} = 0.5$

    –
$$\begin{aligned}\text{Gini}_{\text{root}} &= 1 - (p_{M,\text{root}}^2 + p_{F,\text{root}}^2) = 1 - (0.5^2 + 0.5^2)\\ &= 1 - 0.5 = 0.5.\end{aligned}$$

- Child Node 1: Age < 40 (Yes):
    - Total instances: 4
    - Males (M): 2, Females (F): 2
    - $p_{M,\text{child1}} = \frac{2}{4} = 0.5, \quad p_{F,\text{child1}} = \frac{2}{4} = 0.5$
    -
$$\text{Gini}_{\text{child1}} = 1 - (p_{M,\text{child1}}^2 + p_{F,\text{child1}}^2) = 1 - (0.5^2 + 0.5^2) = 0.5.$$

  Child Node 2.1: age $\geq$ 40 Income $<$ 75000
    - Total instances: 2
    - Males (M): 2
    - $p_{M,\text{child2.1}} = 1, \quad p_{F,\text{child2.2}} = 0$
    -
$$\text{Gini}_{\text{child2.2}} = 1 - (p_{M,\text{child2.2}}^2 + p_{F,\text{child2.2}}^2) = 1 - 1^2 = 0.$$

  Child Node 2.2: age $\geq$, 40 Income $\geq$ 75000
    - Total instances: 4
    - Males (M): 1, Females (F): 3
    - $p_{M,\text{child2.2}} = \frac{1}{4} = 0.25, \quad p_{F,\text{child2.2}} = \frac{3}{4} = 0.75$
    -
$$\begin{aligned}\text{Gini}_{\text{child2.2}} &= 1 - (p_{M,\text{child2.2}}^2 + p_{F,\text{child2.2}}^2)\\ &= 1 - (0.25^2 + 0.75^2) = 0.375.\end{aligned}$$

The overall Gini index is calculated as the weighted average of the Gini indices from each child node, based on the number of instances in each node. We have

$$\text{Overall Gini} = \left(\frac{4}{10} \times 0.5\right) + \left(\frac{2}{10} \times 0\right) + \left(\frac{4}{10} \times 0.375\right) = 0.35.$$

It measures the model's impurity and indicates that there is a 35% chance of incorrectly classifying an instance if it were randomly placed according to the distribution and splits defined in the decision tree model. $\square$

4.5 Issues using synthpop

As indicated in the preceding section, some of the tools incorporated in the `synthpop` package are not appropriate. For example, the Kolmogorov-Smirnov test is designed to test the null hypothesis of tho distributions coming from independent populations. Here, in the package, it is applied to compare original and synthetic data, which are obviously dependent.

In this section we discuss further issues with the package. In particular, the package applies a sequential procedure to generate the $(i+1)$st synthetic variable based on the previously generated i variables. This brings serious questions on how to choose this visiting sequence in the appropriate way. The package does not provide a solution to this problem. To be more specific:

- The selection of the initial variable, the visiting sequence in which variables are chosen, and the design of the predictor matrix usually have a huge influence on the utility of the synthetic data generated, impacting both the marginal distributions and the dependency structures. Before modeling the data set, it is important to analyze the data to verify that the model assumptions are valid. For example, we need to identify which random variables X_i's affect X_j and how to model their dependence.

- In the `synthpop` implementation, it is assumed that there is a sequential dependency, that is, there exists an ordered subset $i_1 < \cdots < i_k$ of $\{1, \ldots, n\}$ such that for all $i \in \{1, \ldots, n\}$ the variable X_i is a function of all the variables $X_{1_i}, \ldots, X_{i_k}$. First, this sequence has to be chosen by the user. The package does not provide any tools to choose the sequence. Furthermore, the existence of such a sequential dependence structure is not guaranteed. In this case, the package may model the relationship inaccurately.

- Compared to other data generation, Machine Learning-type models, such as variational autoencoders (VAEs), generative adversarial networks (GANs), and diffusion models, statistical data generation methods such as those in `synthpop` usually provide statistical inference tools like confidence intervals, hypothesis testing etc. The `synthpop` provides a very limited number of inferential tools and those that are provided do not seem to be implemented correctly; see our discussion on Kolmogorov-Smirnov test. This lack can be a significant drawback when the goal is not only to generate synthetic data, but also to derive statistically robust inferences from the data, which are important for understanding the variability and reliability of the data generated by the model.

In the following examples we illustrate some of the issues mentioned above.

Example 4.5.1. In this example we show an influence of the choice of the sequence in which variables are selected. We generate three variables which are treated as observed data:

- X_1: Generate 1000 observations from an exponential distribution with mean 5.
- X_2: Define X_2 as a linear function of X_1, specifically $X_2 = \frac{1}{2}X_1 + \epsilon$, where ϵ follows a normal distribution with mean 0 and standard deviation 0.1.
- Define X_3 as a linear function of X_1 and X_2, specifically $X_3 = 2X_1 + 3X_2 + \epsilon$, where ϵ follows a normal distribution with mean 0 and standard deviation 0.1.

Using the proposed algorithm, the correct order selected is (X_1, X_2, X_3). Additionally, we consider an incorrect order (X_3, X_2, X_1). We then use the Synthpop package to generate data based on both orders. Notice that if the sequence is wrong, the marginal distribution and the contour plot do not match the original data. But if the sequence is correct, they match pretty well. Recall again that we cannot do a formal hypothesis testing of equality between the marginal distributions, due to dependence. Furthermore, we produce contour plots that indicate that the dependence structure between the variables is not preserved, if the sequence is not chosen in the correct order. We recall that this issue is not addressed in the package at all.

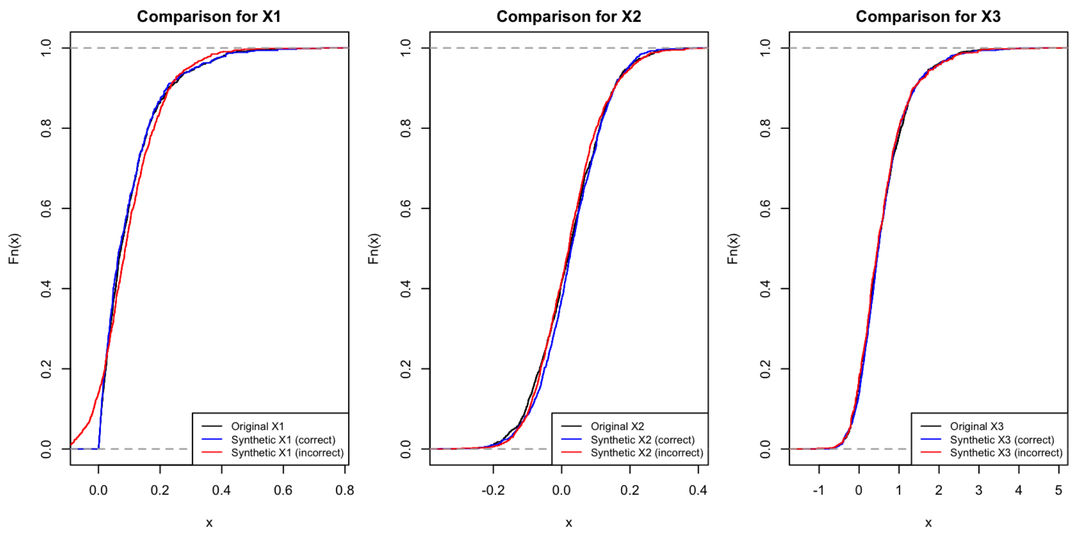


Figure 4.1: Marginal distributions: comparison between the correct and the incorrect sequence

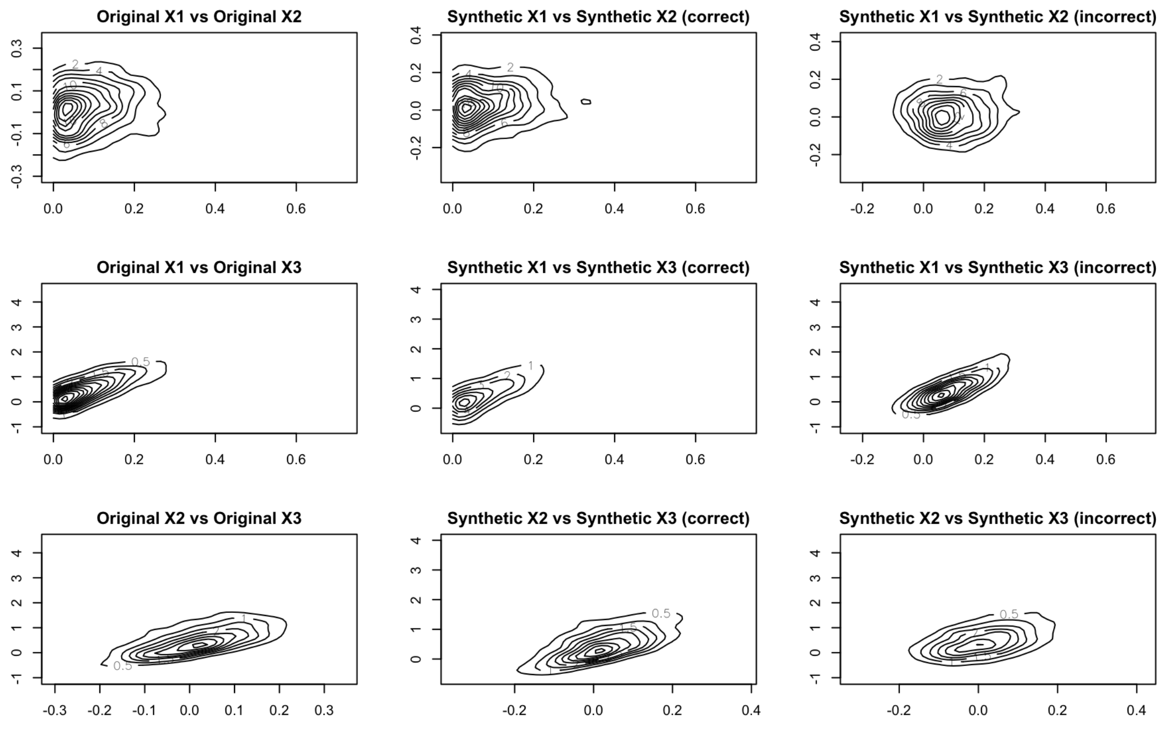


Figure 4.2: Contour plots: comparison between the correct and the incorrect sequence

This experiment shows the importance of choosing variables in the right order when using the `synthpop` package to generate synthetic data. If the order is chosen correctly, the synthetic data look very similar to the original data, both in the marginal distribution and in their dependence structure. This indicates that the relationship between the variables is well maintained. However, when we tried the wrong order (X_3, X_2, X_1), the synthetic data did not match the original well. There are differences in both marginal and dependence structures, which shows that the generated data may not be valid. □

4.5.1 Selecting the sequence

In this section we propose some solutions to the issue of choosing a proper sequence. In what follow, the p-value for the model refers to to the null hypothesis that all parameters in the model are equal to zero, versus the alternative hypothesis that at least one parameter is not equal to zero. That is for paremeters β_i:

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_k = 0$$
$$H_1 : \exists\ \beta_i \neq 0$$

Algorithm 1. In the first algorithm, we choose the proper sequence based on the highest coefficients of determination calculated for potential linear regression models.

- Build regression models f_i for each variable X_i as a function of all other variables X_k, where $k \neq i$. Compute for each i:

  $$X_i = f_i(\{X_k \text{ for all } k \neq i\}).$$

  Choose the model f_{i^*} with the lowest p-value, set the corresponding X_{i^*} aside as the last variable to generate.

- Begin the process to order the remaining variables X_i, $i \neq i^*$:
  - For each remaining variables X_i, model it as a function of all other variables that have not yet been set aside.
  - Continue with the remaining variables:
    * Build regression models:

      $$X_i = f_i(\{X_k \text{ for all } k \neq i \text{ and } X_k \text{ has not been set aside yet}\}).$$

    * Evaluate each model based on its p value. Select the model with the lowest p value and set the corresponding variable aside.
  - Once the sequence of variables $(X_1, X_2, \ldots, X_n)$ is established, use `synthpop` to generate data reflecting this sequence.

The correct correlation between the variables will be maintained if there is an existing a sequential correlation.
The above procedure is described specifically for the linear regression. However, at each step we can choose any other *parametric* regression model, possibly with penalization. We illustrate the above algorithm in the following example.

Example 4.5.2. In this experiment, we generate data from a linear model with three variables: X_1, X_2, and X_3. The relationships are defined as follows: X_2 is linearly dependent on X_1, and X_3 is dependent on both X_1 and X_2. Specifically, we simulate the data with $X_1 \sim \text{Exp}(2)$, $X_2 = 2X_1 + \epsilon$ (with $\epsilon \sim N(0,1)$), and $X_3 = 3X_1 + 5X_2 + \eta$ (with $\eta \sim N(0,1)$).

We apply Section 4.5.1, which iteratively fits linear models and selects the variable with the lowest p-value, stopping when one variable remains. The final sequence is reversed to correctly preserve the dependencies for synthetic data generation. This process assumes a linear model. First, we fit the model for X_1 as a function of X_2 and X_3:

$$X_1 = \beta_0 + \beta_1 X_2 + \beta_2 X_3.$$

The p-value for this model is $p_{X_1} = 4.0446411 * 0.1^{-32}$.
Next, we fit the model for X_2 as a function of X_1 and X_3:

$$X_2 = \beta_0 + \beta_1 X_1 + \beta_2 X_3.$$

The p-value for this model is $p_{X_2} = 1.472311 * 0.1^{87}$.
Finally, we fit the model for X_3 as a function of X_1 and X_2:

$$X_3 = \beta_0 + \beta_1 X_1 + \beta_2 X_2.$$

The p-value for this model is $p_{X_3} = 4.693397 * 0.1^{-93}$.

Since X_3 had the lowest p-value, we selected it as the last variable to fit. We then proceeded by fitting X_1 as a function of X_2 and vice versa. Both had the same p-value of $4.09595 * 0.1^{-17}$, indicating that the order of the last two variables does not matter. In linear regression, the sequence of selecting the final two variables is irrelevant. The resulting sequence is X_3, X_1, X_2, which is the correct sequence for the data. □

The next example illustrates issues when there is no sequential order.

Example 4.5.3. The data is generated as follows:

- X_1: Generate 1000 observations from an exponential distribution with a rate of 0.5.
- X_2: Define X_2 as a linear function of X_1, specifically $X_2 = 2X_1 + \epsilon$, where ϵ follows a normal distribution with mean 0 and standard deviation 0.1.

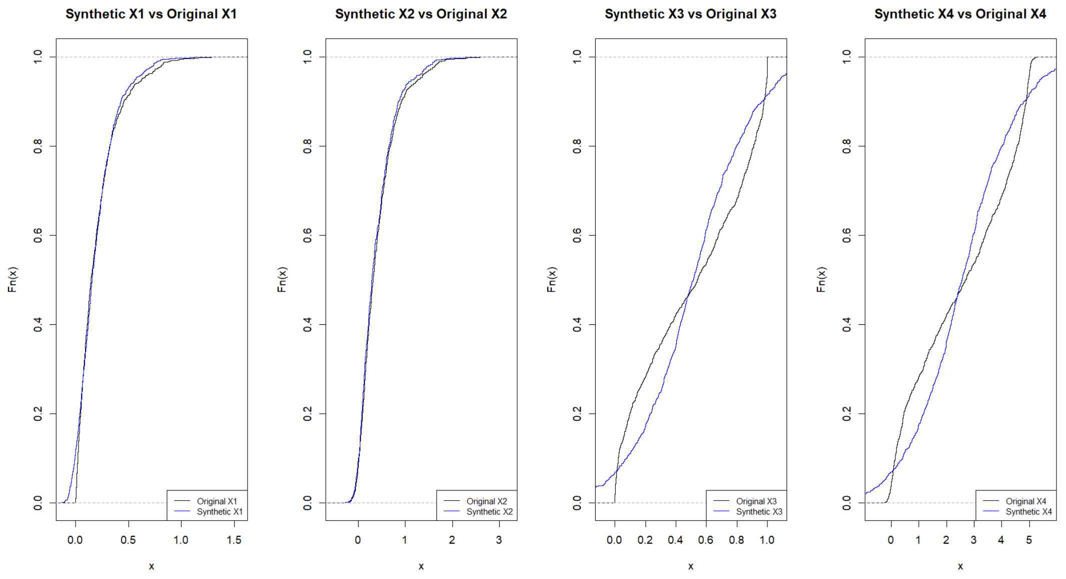


Figure 4.3: Comparison of original and empirical ECDF when there is no sequential relationship

- X_3: Generate independently from a beta distribution with $\alpha = \beta = 0.5$.
- X_4: Define $X_4 = 5X_3 + \epsilon$, where ϵ follows a normal distribution with mean 0 and standard deviation 0.1.

We apply the proposed algorithm Section 4.5.1, which identifies the "correct sequence" as X_2, X_4, X_3, and X_1. Then

- We generate $X_{2,\text{syn}}$ by applying bootstrap to X_2 without replacement.
- Fit a regression model $X_4 = \beta_0 + \beta_1 X_2$, estimate parameters β_i, denoted as $\hat{\beta}_i$, and calculate $X_{4,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_{2\text{syn}}$.
- Fit another regression model where X_3 is a function of X_4 and X_2: estimate the parameters β_i, denoted as $\hat{\beta}_i$, and calculate $X_{3,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_{4,\text{syn}} + \hat{\beta}_2 X_{2,\text{syn}}$.
- Fit another regression model where X_1 is a function of X_4, X_3, and X_2: estimate parameters β_i, denoted as $\hat{\beta}_i$, and calculate $X_{1,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_{4,\text{syn}} + \hat{\beta}_2 X_{3,\text{syn}} + \hat{\beta}_3 X_{2,\text{syn}}$.

Notice that the model assumptions for the generated dataset do not align with the correct assumptions. This mismatch suggests that adjustments or different modeling approaches might be needed to adequately capture the underlying relationships in the data.

Both the marginal distributions and empirical relationships in the synthetic data show clear distortions compared to the original data. These discrepancies suggest that

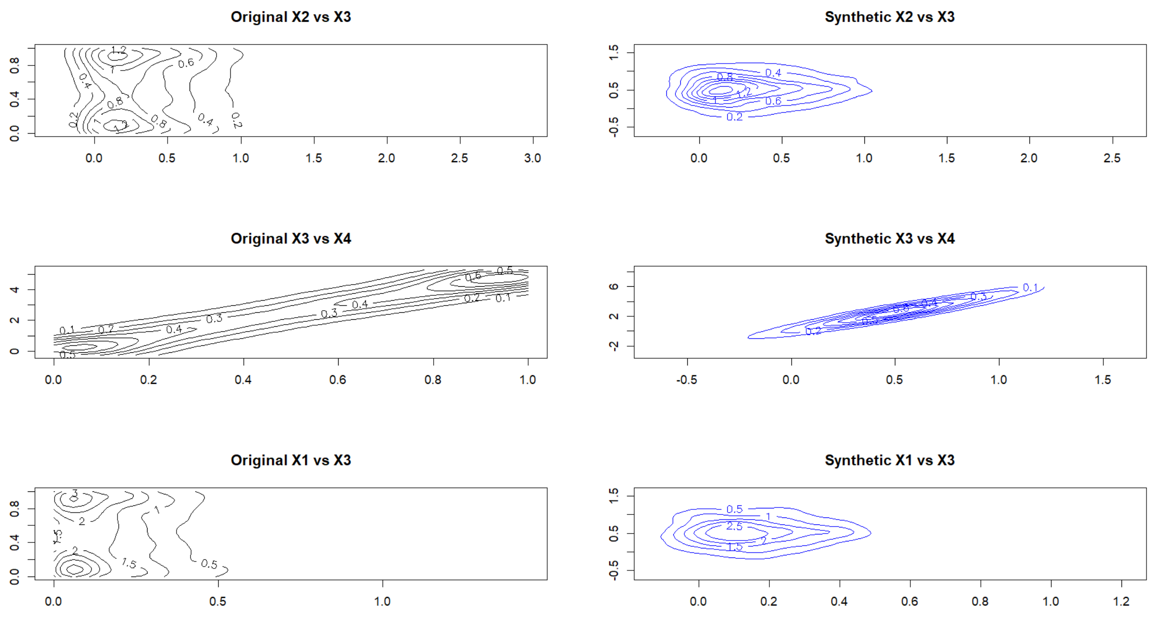


Figure 4.4: Comparison of contour plot when there is no sequential relationship

the synthetic data does not fully capture the original data's structure, impacting its reliability. □

4.5.2 No sequential relationship

- Build a model for each variable X_i as a function of all other variables X_j using observed value, where $j \neq i$. Then we have for each i, we have:

$$X_i = f_{i,\text{temp}}(\{X_{k,obs}, \text{for all } k \neq i\}).$$

- Apply variable selection methods such as Lasso, ridge … to identify and select the most significant variables, denoted by the set K.

- Re-fit the model using only the selected variables from set K. The final model is then:

$$X_i = f_i(\{X_{k,obs} \mid k \in K\}).$$

- Generate synthetic values $X_{i+1,\text{syn}}$ from the model $f_i(X_{k,\text{obs}})$ using frequentist or Bayesian methods to add random noise. This process creates synthetic data for X_k based on observed data from all previously selected variables.

- Repeat the process for all variables in the dataset for each fitted model f_i and we will have $X_{i,\text{syn}}$.
- (Optional) We can then use the generated data to generate another set of synthetic data, denote $X_{i,\text{syn},2}$ where $X_{i,\text{syn},2} = f_i(\{X_{k,\text{syn}} \mid k \in K\})$.

Algorithm 2. The proposed solution addresses the sequence selection problem by removing the need to fit models in an order, preventing errors from one variable affecting the next. The optional step is to ensure that all released data are generated based on synthetic data to prevent information leakage.

Example 4.5.4. We continue with Example 4.5.3 and apply the proposed algorithm 2 using lasso with parameter 0.01 as a variable selection method, We addressed the problem of assuming sequential dependency among variables and successfully identified the correct model: $X_1 = \beta_0 + \beta_1 X_2$, $X_2 = \beta_0 + \beta_1 X_1$, $X_3 = \beta_0 + \beta_1 X_4$, $X_4 = \beta_0 + \beta_1 X_3$. We then proceed as follow,

- Fit a regression model where X_2 is a function of X_1: estimate parameters β_i, denoted as $\hat{\beta}_i$, and compute $X_{2,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_1$.
- Fit a regression model where X_1 is a function of X_2: estimate parameters β_i, denoted as $\hat{\beta}_i$, and compute $X_{1,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_2$.
- Fit a regression model where X_3 is a function of X_4: estimate parameters β_i, denoted as $\hat{\beta}_i$, and compute $X_{3,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_4$.
- Fit a regression model where X_4 is a function of X_3: estimate parameters β_i, denoted as $\hat{\beta}_i$, and compute $X_{4,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_3$.

With the proposed algorithm, the distortions from the original methods are removed, and the generated data follows the same empirical and dependence structure as the original data, returning better accuracy and reliability generated data. □

4.6 Data privacy

In this section, we discuss the privacy of the synthetic data. As introduced in Section 2.5, differential privacy is a measure of privacy protection that guarantees that the inclusion or exclusion of a single individual in the dataset does not significantly affect the outcome of any analysis. We develop an algorithm for differentially private data generation method when fitting regression models. The goal is to generate synthetic data that are differentially private, in the sense of Definition 2.5.3 or Definition 2.5.5.
Recall that our dataset comprises of n observations. The observed data point for the

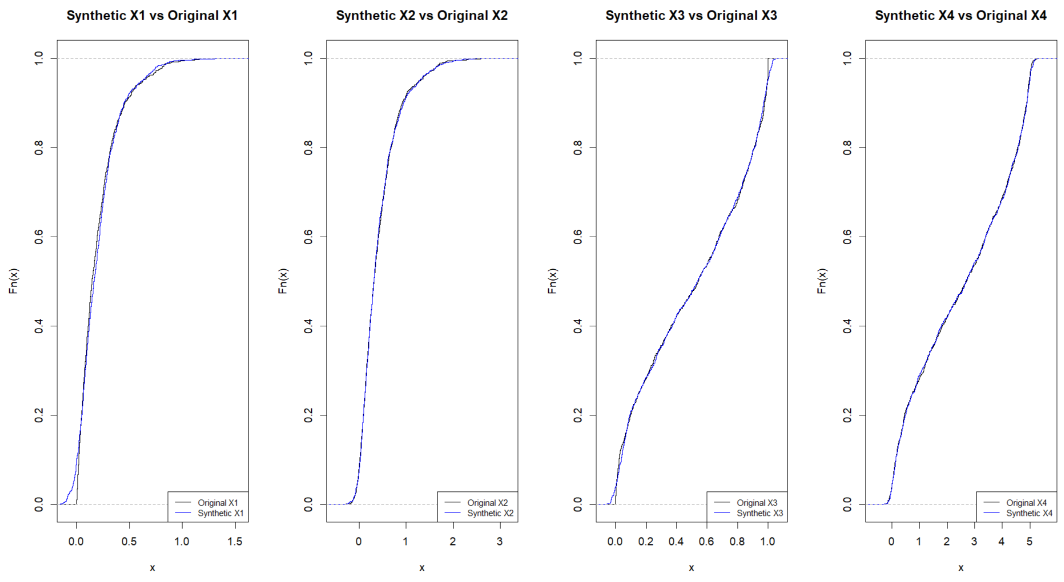


Figure 4.5: Comparison of original and empirical ECDF when there is no sequential relationship

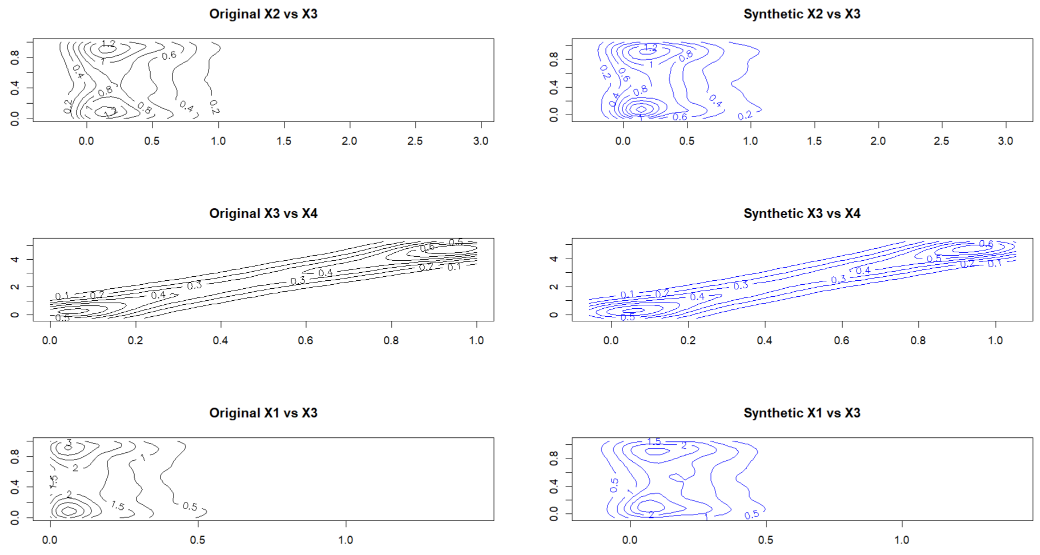


Figure 4.6: Comparison of original and empirical ECDF when there is no sequential relationship

jth individual is denoted by $X_{ij,\text{obs}}$, where the index i represents the ith variable in a multivariate dataset.

To satisfy ϵ differential privacy, the proposed algorithm is as follows:

- We first fit the linear regression model using the selected variables k for each variable i. The fitted model $f_i(X_1, \ldots, X_k)$ is then:

  $$\widehat{X}_i = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \ldots + \hat{\beta}_k X_k.$$

  Therefore, the predicted value for each individual j is

  $$\widehat{X}_{ij} = \hat{\beta}_0 + \hat{\beta}_1 X_{1j,\text{obs}} + \ldots + \hat{\beta}_k X_{kj,\text{obs}}.$$

- We calculate the residuals for the variable i:

  $$X_{ij,\text{obs}} - \widehat{X}_{ij} \ , j = 1, \ldots, n \ .$$

- We calculate the sample variance $\widehat{\sigma}_i^2$ for the residuals of the fitted model.

- For each individual j, we generate a random number E_j from a normal distribution with mean 0 and variance $\widehat{\sigma}_i^2$. Note that we omitted subscript i in the noise variables, since i is fixed.

- For each individual j, we also generate an independent Laplace random noise L_j with mean 0 and scale parameter $\lambda = \frac{\Delta}{\epsilon}$, where Δ is the sensitivity of the identity query applied to the ith variable.

- When generating data $X_{i,\text{syn}}$, we use the fitted model to introduce random noise. That is

  $$X_{i,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_{1,\text{obs}} + \ldots \hat{\beta}_k X_{k,\text{obs}} + E_j.$$

  For jth observed data, we will have:

  $$X_{ij,\text{syn}} = \hat{\beta}_0 + \hat{\beta}_1 X_{1j,\text{obs}} + \ldots \hat{\beta}_k X_{kj,\text{obs}} + E_j = X_{ij,\text{pre}} + E_j.$$

- We then we calculate $d_j = |X_{ij,\text{syn}} - X_{ij,\text{obs}}|$ which is the difference between original data and generated data.

- If $d_j \geq L_j$, we keep the generated value $X_{ij,syn}$, otherwise if $d_j \leq L_j$ we generate $X_{ij,syn} = X_{ij,\text{obs}} + L_j$.

- So, either we use the fitted regression model with E_j noise added, or the original values with the Laplace noise added.

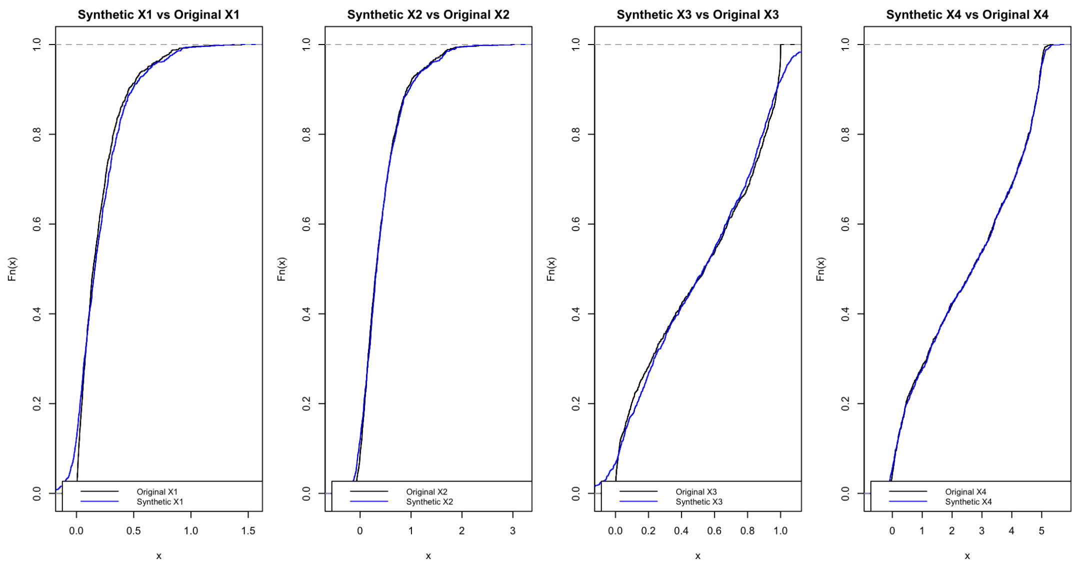


Figure 4.7: Comparison of original and empirical ECDF when there is no sequential relationship

We believe that this proposed algorithm fulfills a vesrion of differential privacy. However, the proof is beyond the scope of the thesis and is a subject of a further research.

Example 4.6.1. We continuous from Example 4.5.4 and apply the algorithm presented above with the level of privacy $\epsilon = 10$. We have:

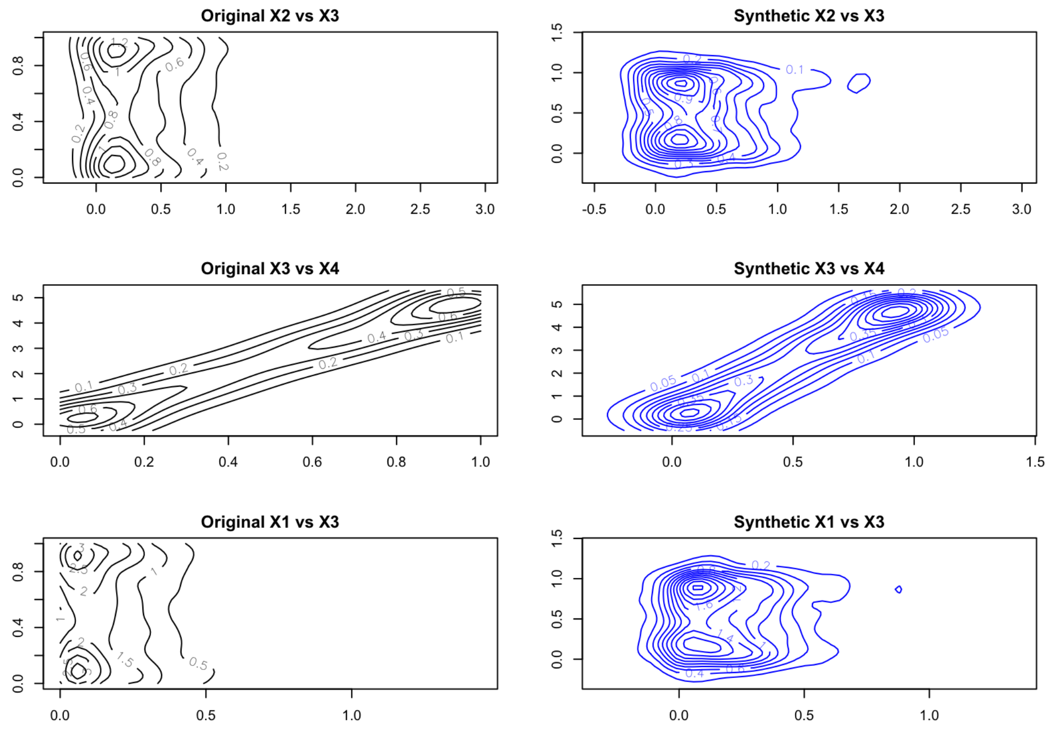


Figure 4.8: Comparison of original and empirical ECDF when there is no sequential relationship

Bibliography

[1] Ian S Abramson. On bandwidth variation in kernel estimates—a square root law. *The Annals of Statistics*, pages 1217–1223, 1982.

[2] David L Banks. Histospline smoothing the bayesian bootstrap. *Biometrika*, 75(4):673–684, 1988.

[3] Jelke G Bethlehem, Wouter J Keller, and Jeroen Pannekoek. Disclosure control of microdata. *Journal of the American Statistical Association*, 85(409):38–45, 1990.

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

[5] Guang Chen and Sallie Keller-McNulty. Estimation of identification disclosure risk in microdata. *Journal of Official Statistics*, 14(1):79, 1998.

[6] Tore Dalenius. Finding a needle in a haystack or identifying anonymous census records. *Journal of Official Statistics*, 2(3):329, 1986.

[7] Cynthia Dwork. Differential privacy. In *International Colloquium on Automata, Languages, and Programming*, pages 1–12. Springer, 2006.

[8] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[9] Khaled El Emam and Fida Kamal Dankar. Protecting privacy using k-anonymity. *Journal of the American Medical Informatics Association*, 15(5):627–637, 2008.

[10] Khaled El Emam, Lucy Mosquera, and Jason Bass. Evaluating identity disclosure risk in fully synthetic health data: Model development and validation. *Journal of Medical Internet Research*, 22(11):e23139, 2020.

[11] Elsayed AH Elamir and Chris J Skinner. Record-level measures of disclosure risk for survey microdata. Technical report, Southampton Statistical Sciences Research Institute, 2004.

[12] Mark Elliot. Final report on the disclosure risk associated with the synthetic data produced by the sylls team. *Report 2015*, 2, 2015.

[13] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[14] Brian V Greenberg and Laura V Zayatz. Strategies for measuring risk in public use microdata files. *Statistica Neerlandica*, 46(1):33–48, 1992.

[15] Peter Hall, Thomas J DiCiccio, and Joseph P Romano. On smoothing and the bootstrap. *The Annals of Statistics*, pages 692–704, 1989.

[16] Diane Lambert. Measures of disclosure risk and harm. *Journal of Official Statistics-Stockholm-*, 9:313–313, 1993.

[17] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2006.

[18] Roderick JA Little. Statistical analysis of masked data. *Journal of Official Statistics*, 9(2):407, 1993.

[19] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.

[20] Iyiola E Olatunji, Jens Rauch, Matthias Katzensteiner, and Megha Khosla. A review of anonymization for healthcare data. *Big Data*, 2022.

[21] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[22] Trivellore E Raghunathan, Jerome P Reiter, and Donald B Rubin. Multiple imputation for statistical disclosure limitation. *Journal of Official Statistics*, 19(1):1, 2003.

[23] Jerome P Reiter and Trivellore E Raghunathan. The multiple adaptations of multiple imputation. *Journal of the American Statistical Association*, 102(480):1462–1471, 2007.

[24] Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[25] Donald B Rubin. Statistical disclosure limitation. *Journal of Official Statistics*, 9(2):461–468, 1993.

[26] Donald B Rubin. *Multiple Imputation for Nonresponse in Surveys*, volume 81. John Wiley & Sons, 2004.

[27] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International, 1998.

[28] Chris Skinner, Catherine Marsh, Stan Openshaw, and Colin Wymer. Disclosure control for census microdata. *Journal of Official Statistics-Stockholm-*, 10:31–31, 1994.

[29] Chris J Skinner and Mark J Elliot. A measure of disclosure risk for microdata. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 64(4):855–867, 2002.

[30] Joshua Snoke, Gillian M Raab, Beata Nowok, Chris Dibben, and Aleksandra Slavkovic. General and specific utility measures for synthetic data. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 181(3):663–688, 2018.

[31] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):571–588, 2002.

[32] Jennifer Taub, Mark Elliot, Maria Pampaka, and Duncan Smith. Differential correct attribution probability for synthetic data: An exploration. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2018, Valencia, Spain, September 26–28, 2018, Proceedings*, pages 122–137. Springer, 2018.

[33] Leslie Taylor, Xiao-Hua Zhou, and Peter Rise. A tutorial in assessing disclosure risk in microdata. *Statistics in Medicine*, 37(25):3693–3706, 2018.

[34] Matthias Templ. Statistical disclosure control for microdata. *Cham: Springer*, 2017.

[35] George R Terrell and David W Scott. Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265, 1992.

[36] Leon Willenborg and Ton De Waal. *Elements of Statistical Disclosure Control*, volume 155. Springer Science & Business Media, 2012.

[37] Mi-Ja Woo, Jerome P Reiter, Anna Oganian, and Alan F Karr. Global measures of data utility for microdata masked for disclosure limitation. *Journal of Privacy and Confidentiality*, 1(1), 2009.

Index

Appendix

Probability distribution

Laplace Distribution. The Laplace distribution, is defined by its probability density function (PDF):

$$f(x \mid \mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right),$$

where μ denotes the location parameter, representing both the mean and the median, and b is the scale parameter that determines the distribution's spread.

Distances

Euclidean Distance. The Euclidean distance between two points $\mathbf{x}$ and $\mathbf{y}$ in $\mathbb{R}^n$ is:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

Kullback-Leibler Divergence. The Kullback-Leibler divergence from a probability distribution Q to a probability distribution P measures the information gained by transitioning from the prior distribution Q to the posterior distribution P. It is quantified by:

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx,$$

where $p(x)$ and $q(x)$ are the PDFs of distributions P and Q, respectively. This measure is asymmetric, i.e., $D_{\mathrm{KL}}(P \parallel Q) \neq D_{\mathrm{KL}}(Q \parallel P)$.

Hamming Distance. The Hamming distance is a metric used to measure the difference between two strings of equal length. It is defined as:

$$D_{\mathrm{H}}(a, b) = \sum_{i=1}^{n} 1[a_i \neq b_i],$$

where $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$.