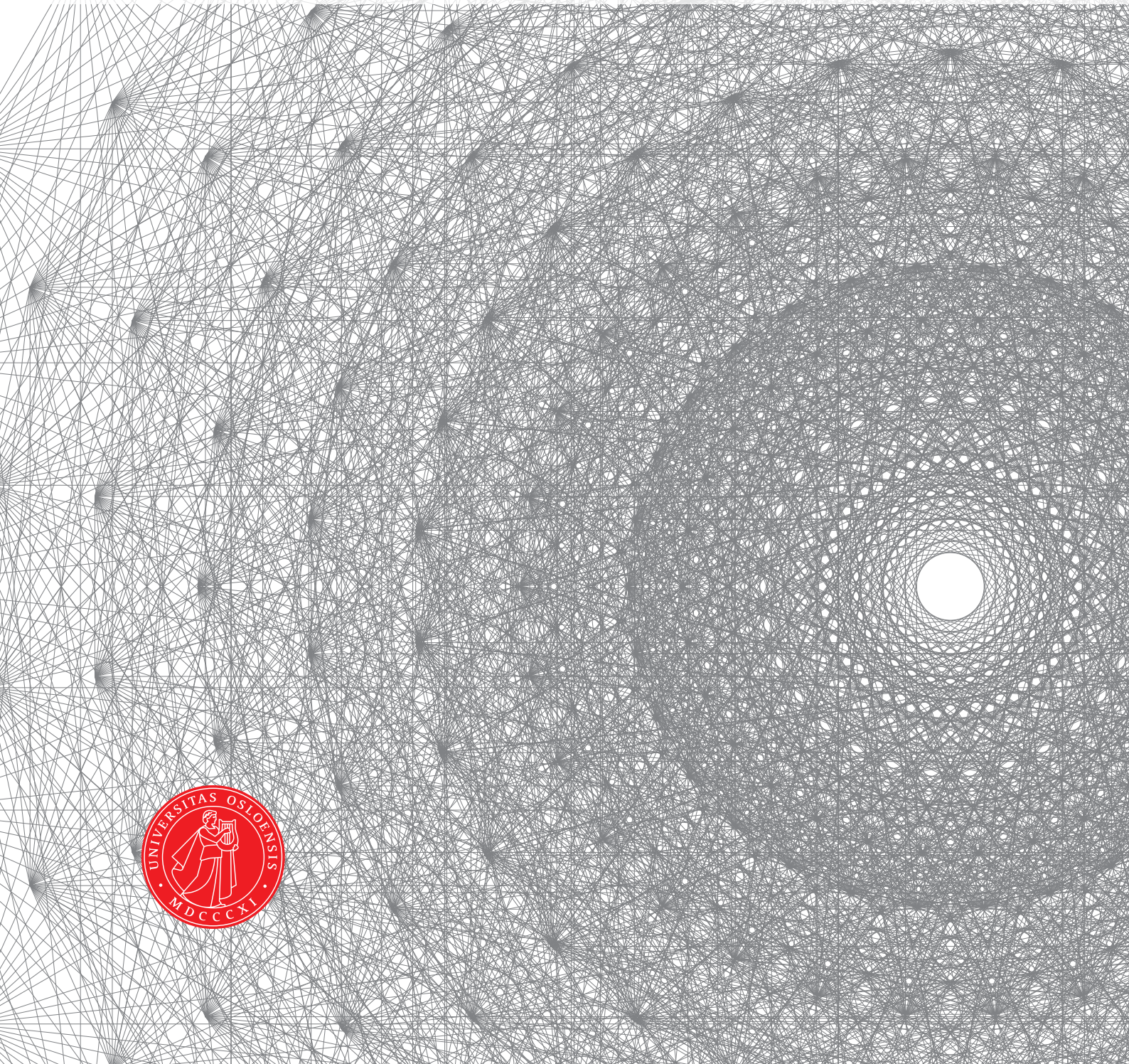
UiO : Department of Mathematics
University of Oslo

Privacy and Utility Evaluation of Synthetic Data

for Multi-State Time-to-Event Applications

Ingvild Riiser

Master's Thesis, Autumn 2023



This master's thesis is submitted under the master's programme *Stochastic Modelling, Statistics and Risk Analysis*, with programme option *Statistics*, at the Department of Mathematics, University of Oslo. The scope of the thesis is 60 credits.

The front page depicts a section of the root system of the exceptional Lie group E_8, projected into the plane. Lie groups were invented by the Norwegian mathematician Sophus Lie (1842–1899) to express symmetries in differential equations and today they play a central role in various parts of mathematics.

Abstract

Synthetic data has gained attention over the last years because of its ability to safeguard the privacy of real data points while still ensuring data utility. These properties are beneficial in many domains and sectors working with sensitive data, particularly to public agencies, which govern large amounts of data on individuals. Most previous works on synthetic data centres around tabular data, and while some research has been done on synthetic survival data, the topic of synthetic multi-state time-to-event (MS-TTE) data has yet to be considered. In this thesis, we develop a novel semi-parametric approach to synthesising MS-TTE data, which combines a non-parametric tabular synthesiser with a parametric multi-state survival regression model. We use Weibull regression and both clock-reset and clock-forward models. Moreover, we extend our approach into an MS-TTE model with a differential privacy guarantee. We also introduce a novel differentially private Weibull regression model.

We review selected evaluation methods for synthetic data in terms of privacy and utility evaluation. The standard approach evaluates synthetic data based on a single data set, which does not account for the variance between synthetic data sets generated from the same synthesiser. We propose a distance-based evaluation framework which adjusts for this variance.

Using an open-access data set, we demonstrate our proposed synthesisers for MS-TTE data with and without differential privacy. Furthermore, we exemplify the evaluation of these synthesisers and their synthetic data by adapting reviewed methods to an MS-TTE setting and utilising our proposed evaluation framework.

Acknowledgements

First and foremost, my sincere gratitude goes to my supervisors, Ingrid Kristine Glad and Anders Løland. I greatly appreciate all the feedback and suggestions and your encouragement and enthusiasm. Every meeting with you left me feeling uplifted and inspired. Further, thank you to my co-supervisor, Robindra Prabhu at NAV, for introducing me to the topic of synthetic data and for your insightful comments and ideas.

To my friends and fellow students from various disciplines, thank you for all the valuable discussions and much-appreciated lunch breaks. Special thanks to Lisabeth Toft Tomter for help with proofreading.

A thanks is also owed to Brian Eno for accompanying my thoughts and keeping me focused over the last two years. Although, I wonder if I can listen to Apollo ever again.

I am grateful for all the love and support I have received from my friends and family during this time. Most of all, thank you to Elise for everything.

Contents

List of Acronyms

AI	Artificial intelligence.
AUROC	Area under the receiver operating characteristic.
CGAN	Conditional GAN.
CPAR	Conditional Probabilistic Auto-Regressive.
CTGAN	Conditional Tabular GAN.
DCR	Distance to closest record.
GAN	Generative adversarial network.
IOU	Interval overlap utility.
kNN	K-nearest neighbours.
KS	Kolmogorov-Smirnov.
MCMC	Markov chain Monte Carlo.
MIA	Membership inference attack.
MLE	Maximum likelihood estimate.
MS-TTE	Multi-state time-to-event.
NAV	The Norwegian Labour and Welfare Administration.
NDPA	Norwegian Data Protection Authority.
NNDR	Nearest neighbour distance ratio.
NSAI	National Strategy for Artificial Intelligence.
PDF	Probability density function.
RNN	Recurrent neural network.
SDC	Statistical disclosure control.
SDU	Standardised difference utility.
Tenor	Test Norway.

TRTR	"Train on real, test on real".
TSTR	"Train on synthetic, test on real".
TSTS	"Train on synthetic, test on synthetic".
VAE	Variational autoencoder.
WPHR	Weibull proportional hazards regression.

CHAPTER 1

Introduction

1.1 Background and Motivation

In 2020, the Norwegian Government published a National Strategy for Artificial Intelligence (NSAI) (Norwegian Ministry of Local Government and Modernisation, 2020), in which they presented a framework for the development and use of artificial intelligence (AI) within the public and private sector. Motivated by the possibilities of increasing efficiency and improving services, public agencies are encouraged to actively explore the possibilities of AI. The Norwegian Labour and Welfare Administration (NAV) established an AI-lab in 2017 and has since worked on the development of responsible AI. Responsible AI covers aspects such as fairness, explainability and privacy (Vidnes Jensen and Pihl Lyngstad, 2019).

Data is essential for developing AI and machine learning models. NAV administers more than one-third of the Norwegian National Budget through welfare benefits such as child benefits, sickness benefits and pensions (NAV, 2019). Through these services, NAV collects personal data on most citizens at multiple stages of their lives. Data on unemployment and sickness can be of a sensitive nature. Thus, ensuring the their users' privacy is a central challenge in NAV's development of responsible AI. Using anonymised data to train models is a possible and commonly used solution. However, with re-identification attacks getting increasingly sophisticated, this can no longer be regarded as a satisfactory method for protecting the privacy of individuals in the data (Rocher, Hendrickx and Montjoye, 2019).

Another challenge of using personal data is the requirement of informed consent for each purpose, which can be a lengthy process and can introduce consent bias in the data (El Emam, 2020). Data collected for the public sector is often excepted from these regulations through specific statutory provisions (Norwegian Ministry of Local Government and Modernisation, 2020). However, because they were written prior to the development of today's technological advancements in AI, further clarifications are necessary (Norwegian Data Protection Authority, 2022). In a regulatory sandbox project between NAV and the Norwegian Data Protection Authority (NDPA), NAV presented a predictive model based on personal data. NDPA concluded that using the model was allowed within the current legislation. Nevertheless, it was unclear if personal data could legally be used as training data for such a model (Norwegian Data Protection Authority, 2022).

NSAI proposes synthetic data as a possible solution to the challenges of personal data. Synthetic data with sufficient utility can be used instead of real data in developing machine learning models. Another benefit is that synthetic data can, in principle, be openly shared within and outside the agency, encouraging more innovation and research (Norwegian Ministry of Local Government and Modernisation, 2020).

In practice, synthetic data still poses a privacy risk (Stadler, Oprisanu and Troncoso, 2022), and the legal debate around the use of synthetic data is in its early stages. This discussion is naturally outside the scope of this thesis, and we refer the reader to Bellovin, Dutta and Reitinger (2019) and Gal and Lynskey (2023) for analyses of the legal implications of synthetic data. Even so, the legal aspect of synthetic data highlights the need for thorough quantitative evaluations and risk assessments of the privacy of synthetic data.

This master project was induced by NAV's aspirations to explore synthetic

data and how they can use it within their AI and machine learning research. However, this thesis aims for a more general scope and can be relevant to many sectors and industries.

1.2 Problem Statement

In this thesis, we will explore the novel concept of synthetic multi-state time-to-event (MS-TTE) data, which requires different methods in terms of synthesis and evaluation than common data types like synthetic tabular, text and image data. MS-TTE data can be used to model how individuals move between states over time. NAV data has previously been used to analyse individuals' transitions between work, partial and full sick leave, work assessment allowance and disability pensions through multi-state models (Gran et al., 2015). With synthetic MS-TTE data, such studies can more easily meet privacy requirements.

MS-TTE data is a type of survival data, and limited work has been done on synthetic survival data thus far. During the work on this thesis, Norcliffe et al. (2023) published *SurvivalGAN*, a machine learning model for synthesising survival data. However, it does not provide a method for synthesising MS-TTE data. To the best of our knowledge, this data type has not been discussed in a synthetic data context before. Therefore, we develop novel MS-TTE synthesisers based on existing parametric survival regression methods and tabular synthesisers. Then, we discuss how existing and proposed privacy and utility evaluation methods should be applied to this data type. The main objective of this thesis is not to create a complex model that generates synthetic data with perfect utility, which is to be used interchangeably with real data. Instead, we focus on providing a starting point for MS-TTE synthesis and utilise our proposed MS-TTE synthesiser models to illustrate various evaluation measures.

1.3 Chapter Outline

Chapter 2 offers an introduction to the field of synthetic data. We present the definitions and terms used in this thesis and discuss possible use cases. Next, we offer a short survey of existing work on the types of synthetic data we will encounter, namely tabular, sequential and survival data. Furthermore, we review a selection of utility evaluation metrics used for synthetic data. Different types of privacy disclosure applied to a synthetic data setting are discussed, and we consider the privacy challenge to membership inference attacks (MIAs), for which we provide a novel variant. Moreover, we introduce differential privacy and discuss how it can be applied to synthetic data. Finally, we explain how proximity measures can be used to evaluate both the privacy and utility of synthetic data. Our main contribution in this chapter is a novel evaluation procedure which utilises existing distance metrics and repeated synthesis steps for evaluating a synthesiser and not simply a specific synthetic data set.

In **Chapter 3**, we provide the requisite theoretical framework for survival analysis and multi-state models that is needed for the remainder of this work. We discuss survival regression models and Weibull regression in particular. Further, we consider censoring and time inhomogeneous multi-state models. Our first major contribution in this chapter is an algorithm which combines tabular

synthesis and multi-state simulation, which can generate synthetic MS-TTE data. We propose both clock-forward and clock-reset MS-TTE synthesisers. The chapter concludes with a discussion of differential privacy for survival data in general and MS-TTE data in particular. We expand on prior work by providing a novel differentially private Weibull regression model. Our second main contribution is a differentially private multi-state model.

We demonstrate the generation of an MS-TTE synthetic data set in **Chapter 4** using an open-access data set. First, we show how naive synthesis methods fall short for MS-TTE applications. Then, we use our proposed methods to generate synthetic data sets with both clock-forward and clock-reset properties. We evaluate the data using the evaluation procedures described in Chapter 2. Finally, we carry out an MIA targeting the synthetic data.

Chapter 5 picks up the differentially private MS-TTE model presented in Chapter 2, which we use to generate synthetic data with a differentially private guarantee, using the same example data set as in Chapter 4. We show how the hyper-parameters can be tuned and evaluate the differentially private synthetic data in terms of privacy and utility.

We conclude and provide directions for further work in **Chapter 6**. In **Appendix B**, we provide an overview of the source code, which is available on GitHub.

CHAPTER 2

Synthetic Data

2.1 Introduction to Synthetic Data

2.1.1 Early Developments

Fully synthetic data was first proposed by Rubin (1993) as a way to protect tabular microdata against privacy attacks. He proposed to release synthetic data in place of sensitive real data and performed synthesis based on multiple imputation methods for missing values (Drechsler, 2011; Rubin, 1987). For a cost of higher variance in the synthetic data, this method protected the privacy of real individual points, as none of the synthetic data points were mapped directly from single real points. Synthetic data was presented as a more radical approach than other statistical disclosure control (SDC) methods used at the time, such as record swapping, adding random noise and other data masking techniques (Drechsler, 2011; Raab, Nowok and Dibben, 2016).

2.1.2 A Tailored Definition

Synthetic data has evolved in numerous directions over the last three decades. Today, the term can be used to describe all sorts of data types, such as generated text, audio and images, as well as tabular, sequential and survival data (Creswell et al., 2018; Jordon, Szpruch et al., 2022; Norcliffe et al., 2023). To define such a broad term is a challenging task, and the literature on synthetic data offers various definitions. We paraphrase three of them in Definition 2.1.1, Definition 2.1.2 and Definition 2.1.3, and we wish to discuss their similarities, where they differ and possible limitations before we present our own definition.

Definition 2.1.1 [Synthetic data according to Duncan, Elliot and Salazar-González (2011)]

Synthetic data is data intended to be released as a replacement for real data. It is generated from a probabilistic model fitted to this real data, whose purpose is generating synthetic data which can be of general use.

Beginning with Definition 2.1.1, a probabilistic model allows for non-direct linkage between the real input data and synthetic output data of the model, as there is a probability attached to any given input and output combination (Duncan, Elliot and Salazar-González, 2011). This is analogous to a randomised algorithm, as defined by Dwork and Roth (2014, Defintion 2.2). Both parametric and non-parametric models can be used as probabilistic models (Drechsler and Reiter, 2011). Because probabilistic models can return many different outputs based on one input signal, it means that we can generate as much synthetic data as we wish, regardless of the size of the real input data. A possible limitation of Definition 2.1.1 is that it makes little distinction between synthetic and simulated data since the latter is also generated from a model that has often been fitted to real data. The two terms are sometimes used interchangeably (Beaulieu-Jones et al., 2019).

Definition 2.1.2 [Synthetic data according to Jordon, Szpruch et al. (2022)]

Synthetic data is generated by means of a purpose-built mathematical model or algorithm, with the aim of using the data to solve data science tasks.

Definition 2.1.2 is similar to the definition of Duncan, Elliot and Salazar-González in the sense that the synthetic data should be generated from a model built for that purpose, which we argue can be too limiting. A wide range of models can easily be extended into synthetic data generators. To illustrate, say that a linear regression model is fitted to real data with the purpose of performing inference. If we intend to use it to generate synthetic data, we can begin by sampling from the (joint) distribution of the covariate(s). Next, the response variable is generated by applying the estimated regression model to the sampled covariates and adding a random noise term. Combined, we then have a synthetic data set consisting of the covariate(s) and a response variable. Another example is autoencoders, which are composed of an encoder and a decoder. They can be used to compress and restore data, but the decoder can double up as a data generator (Spinner et al., 2018). One can argue that the definition by Jordon, Szpruch et al. still hold, as both models need adjustments[1] before they can be used to create synthetic data. In the regression case, we say that the regression model is the original model, and the adjusted model is the combined covariate sampler and regression model. Therefore, one may claim that the adjusted model is purpose-built for synthesis. However, we seek a definition that avoids any such confusion by not enforcing unnecessary restrictions on the type of models used.

Definition 2.1.3 [Synthetic data according to Templ (2017)]

Templ's definition calls for synthetic data of a certain quality. Firstly, the real and synthetic data should be close to equal in terms of distribution, correlation structure and heterogeneity between subgroups. Secondly, the synthetic data should ensure data confidentiality of the real data, and synthetic data points should not be directly mapped from unique real data points.

In Definition 2.1.3, Templ defines synthetic data in terms of its quality. He is not, like Duncan, Elliot and Salazar-González and Jordon, Szpruch et al., concerned with the model used to generate the synthetic data. The two criteria refer to the concepts of data privacy and utility (El Emam, 2020), which are subject to further discussion in Section 2.3. Duncan, Elliot and Salazar-González and Jordon, Szpruch et al. also comment on the utility of the synthetic data, as they require that the synthetic data is useful in a general sense and for data science tasks, respectively. However, they do not place specific statistical requirements, as we see in Definition 2.1.3. Furthermore, as opposed to Duncan, Elliot and Salazar-González, we note that Definition 2.1.3 separates between synthetic and simulated data since the latter is generally not required to meet privacy and utility requirements.

[1] Standard autoencoders often perform poorly at generating data (Nikolenko, 2021), and Variational autoencoders (VAEs) (Kingma and Welling, 2014), which are mostly used as synthetic data generators (Figueira and Vaz, 2022; Nikolenko, 2021), underperform on data compression tasks (Spinner et al., 2018).

Based on these definitions, we create our own that aims to be both general and suitable for the scope of this thesis. As discussed, we do not wish to limit the synthesiser model to be purpose built, as Definition 2.1.1 and 2.1.2 require. Nevertheless, we shall explicitly state that the purpose of the synthetic data is to substitute real data, which is in line with Definition 2.1.1. We prefer to have a clear distinction between synthetic and simulated data, which is why we choose to include requirements of privacy and utility, like in Definition 2.1.3. The terms privacy and utility ought to be used explicitly in the definition, as they are widely used terms for the evaluation of the synthetic data (Bellovin, Dutta and Reitinger, 2019; El Emam, 2020; Snoke et al., 2018).

Definition 2.1.4 [Synthetic data]

A synthetic data set $\mathcal{D}_s$ is generated from a synthesiser $\mathcal{S}$ with the purpose of substituting some real data set $\mathcal{D}_r$. $\mathcal{S}$ is fitted to $\mathcal{D}_r$ and is used to generate synthetic data with satisfactory utility with respect to the intended use, but without a concerning cost of data privacy of individual data points in $\mathcal{D}_r$.

We state that the purpose of $\mathcal{D}_s$ is to substitute $\mathcal{D}_r$, meaning that $\mathcal{D}_s$ can be used to perform the same tasks in a similar fashion to $\mathcal{D}_r$. The utility is a measurement of how well $\mathcal{D}_s$ performs at these tasks compared to $\mathcal{D}_r$, and can refer to both statistical similarity, like in definition **three**, and performance of specific machine learning tasks, like in definition **two**. Definition 2.1.4 is deliberately vague by not specifying what is regarded as a satisfactory utility or a concerning privacy cost. These bounds should be set in accordance with the purpose of the synthetic data, which we discuss in Section 2.1.3. Additionally, synthetic data with poor data privacy or low utility should still be regarded as synthetic data, even when the quality is dubious.

If we assume that $\mathcal{D}_r$ is sampled from a true model, $\mathcal{T}$, we can think of $\mathcal{S}$ as an estimate of $\mathcal{T}$. If $\mathcal{S} = \mathcal{T}$, then $\mathcal{D}_r$ and $\mathcal{D}_s$ are independent samples from the same model (Duncan, Elliot and Salazar-González, 2011). Raab, Nowok and Dibben (2016) refer to this as the Synthesising Distribution Assumption. This guarantees a $\mathcal{D}_s$ with full utility and full privacy, which would be an ideal synthetic data set. However, independence between the two sets $\mathcal{D}_r$ and $\mathcal{D}_s$ is not achievable in practice since $\mathcal{T}$ is unknown (and will likely not exist in practice), and $\mathcal{S}$ is fitted to $\mathcal{D}_r$. Thus, we need a quantifiable evaluation of the privacy and utility of $\mathcal{D}_s$. This evaluation can be included in $\mathcal{S}$, but can also be implemented as post-processing steps. We continue this discussion in Sections 2.3-2.6.

Here, we use the term synthesiser to refer to the synthetic data generator, while generator is also commonly used. In Definition 2.1.5, we propose a definition of a synthesiser $\mathcal{S}$ that holds in the general case. Based on Definition 2.1.1, which refers to probabilistic models, the architecture of the generators of GANs[2] (Creswell et al., 2018) and partly inspired by the notation of a mechanism in Dwork and Roth (2014), we present the following definition:

[2]Generative adversarial networks (GANs) take a random noise input.

Definition 2.1.5 [Synthesiser]

A synthesiser, $\mathcal{S}$, is a probabilistic that generates a synthetic data set $\mathcal{D}_s$. It is fitted to a real data set $\mathcal{D}_r$ containing m data points, each with domain $\mathcal{X}$. Given any random number generator on $\mathbb{R}$, $\mathcal{S}$ can generate n synthetic data points from $\mathcal{X}$.

$$\mathcal{S} : \mathbb{R} \to \mathcal{X}^n.$$

A shortcoming of Definition 2.1.5 is the exclusion of augmented synthetic data, which can be used to create synthetic data points of a larger domain than the real data points. For instance, augmented generated image data can include captions, even when the real training data does not (Jordon, Szpruch et al., 2022). We use Definition 2.1.5 since data augmentation is outside the scope of this thesis. Note that some synthesisers, such as autoencoders, require that the noise signal is a vector so that $\mathcal{S}$ maps from $\mathbb{R}^k$, where $k \in \mathbb{N}$.

2.1.3 General Use Cases

In Definition 2.1.4, we state that the role of synthetic data is to substitute real data. The distinction between a substitute and a replacement is important because the synthetic data should not be treated as a perfect equivalent of the real data. Breugel, Qian and M. v. d. Schaar (2023) demonstrate how a naive approach to synthetic data, namely using it as if it was real, causes inaccurate results when the synthetic data is applied to various tasks. This is a result of inadequate utility. The level of privacy will also depend on the purpose. For instance, data shared internally within an organisation or company will require less privacy than externally shared data. In this section, we will discuss a selection of general use cases and their requirements for privacy and utility.

Test data

Synthetic data is often used for testing purposes, like tests of information systems. In scenarios where external partners execute tests or when the development process requires a constant supply of test data, synthetic test data is especially useful. This is because the process of getting access to real data can be tedious (Behjati et al., 2019; El Emam, 2020). In other cases, synthetic data is used as test data because there is not enough real data available (El Emam, 2020).

Behjati et al. (2019) discuss how it can be useful for several public agencies and administrations to have access to the same test data since many systems rely on a data flow between different entities. This can enable more complex testing scenarios. Test Norway (Tenor) data[3] is such a system maintained by the Norwegian Tax Administration. The system is developed for testing purposes and contains a synthetic version of the Norwegian population. Tenor data can be used in Dolly, which is NAV's own system for generating synthetic data for specific testing purposes (NAV, 2022).

[3]Norwegian: Test-Norge.

In summary, the level of privacy of the synthetic test data depends on whether the testing is performed internally or externally. In terms of utility, the synthetic test data is generally not required to closely follow the same distribution as the real data, but they need to contain enough outliers so that unusual events are covered (El Emam, 2020). Moreover, it is important that the synthetic data has the same data structure as the real data (Jordon, Szpruch et al., 2022), meaning that they have the same columns and data types of each column. Because of this, data that is generated by a theoretical model can be used (El Emam, 2020). A theoretical model can be interpreted as a prior distribution, and it represents believes about the population without looking at the real data. Because the theoretical model is not trained on real data, such data cannot be described as synthetic data in accordance with Definition 2.1.4. Instead, we call it simulated data.

Machine learning and AI

Synthetic data is used for both model selection and model training. A typical model selection task consists of finding the most suitable model for a given task, say classification, among a set of candidates (Breugel, Qian and M. v. d. Schaar, 2023; Jordon, Szpruch et al., 2022). Since this is typically done in an exploratory phase, permissions for accessing the real data may not yet be in place, which makes synthetic data a useful substitute. Through a single test-train split of the synthetic data or cross-validation, the model with the smallest test error is selected. Breugel, Qian and M. v. d. Schaar (2023) perform such an experiment and conclude that synthetic data sets tend to favour models with similar structures as their synthesisers rather than the model with the best result on real data. They used a complex deep learning based synthesiser[4], and their conclusion does not necessarily generalise to less complex synthesisers.

Example 2.1.6 [Using synthetic data for model selection]

Picture a common scenario where an analyst wants to fit a predictive model to a set of training data. The training data contains many covariates, but only some of them are significant to the response variable. The significant covariates are not known beforehand, and all the covariates need to be included in the model selection process. However, the analyst is not granted access to all the variables, due to privacy constraints. Suppose the model selection process can be carried out on a synthetic data set instead. In that case, the analyst can use synthetic data to find the subset of the covariates that are significant. The training data containing these covariates are then handed over to the analyst for the training of the final model.

After the model selection, the synthetic data may also be used to fit the model. However, Snoke et al. (2018) find in their review that while it is conventional to use synthetic data for model exploration and selection, final models are fitted to real data. Jordon, Szpruch et al. (2022) suggest that synthetic data should only be used for the pre-training of a model, meaning that real data should be used to tune the final model.

[4] CT-GAN

These views represent a more careful and conservative approach to synthetic data, and more optimistic views exist. A much cited prediction by the tech research and consulting company Gartner is that "by 2030, synthetic data will completely overshadow real data in AI models" (Laurence Goasduff, 2022). This is a bold claim, in our opinion. The "garbage in, garbage out" principle means that if the utility of the synthetic data is lacking, then the resulting model will also have low utility. This is a current issue, because as the GPT models have gained popularity, an increasing amount of text data on the internet has become synthetic. Large language models (LLMs) are trained on vast amounts of text data from the internet, making it challenging to filter out synthetic text from the training material. An increasing amount of synthetic text in the training data can cause weaker models (Shumailov et al., 2023).

We argue that machine learning is an important use case for synthetic data, but it is important that the synthetic data used for training is clearly marked as such, so the confidence in the results can be adjusted accordingly. Esteban, Hyland and Rätsch (2018), Xie et al. (2018) and Zhao et al. (2021) all demonstrate how synthetic data can be used to fit a final model for supervised learning tasks.

Satisfactory utility in this context means that the synthetic data will reach the same conclusion as the real data when performing model selection and that a model fitted to the synthetic data is close to the same as a model fitted to the real data. As we have discussed, this can be difficult to obtain. We will return to a discussion on how this can be assessed in Section 2.4.

The amount of privacy necessary will again depend on how the synthetic data will be distributed. It is important to note that this does not only apply to the data itself, but also to the distribution of any model fitted to the synthetic data. This is due to linkage and membership inference attacks (Stadler, Oprisanu and Troncoso, 2022).

Statistical inference

Synthetic data used for statistical inference tasks such as hypothesis testing, point and interval estimations must be of high utility. Even so, the conclusions will be weaker than compared to using real data, because of added noise, which may require hypothesis tests to have adjusted significance levels (Jordon, Szpruch et al., 2022) and increased confidence intervals (Drechsler, 2011). Raab, Nowok and Dibben (2016) derive several estimates of the variance of point estimates from synthetic data, but the results only hold for large samples of real data and where the synthetic data meets the Synthesising Distribution Assumption (Raab, Nowok and Dibben, 2016) discussed in Section 2.1.2. Since this is often not the case, Raab, Nowok and Dibben (2016) strongly discourage using synthetic data for any final inferential analysis.

2.2 A Short Survey of Selected Synthetic Data Types and Synthesisers

In this section, we present a short survey of synthetic data types relevant to this thesis and examples of previous work on synthesisers tailored to each data

type. While not an exhaustive list, this serves as an introduction to topics we will return to in later chapters.

2.2.1 Tabular Data

Tabular data is stored in two-dimensional tables, where generally each column represents a variable and each row is an observation. Because this is a particularly common data structure, tabular synthetic data is a heavily researched field (Hernandez et al., 2022; Kiran and Kumar, 2023; Rajabi and Garibay, 2022; Sauber-Cole and Khoshgoftaar, 2022; L. Xu, Skoularidou et al., 2019; Zhao et al., 2021).

There are several challenges to synthetic tabular data generation that do not apply to simpler data types such as image generation. For image data, the pixel values follow a normal distribution, whereas tabular variables can follow more complex distributions (L. Xu, Skoularidou et al., 2019). Other challenges include mixed data types, with both continuous and discrete variables, multi-modal distributions and imbalanced discrete data (L. Xu, Skoularidou et al., 2019; Zhao et al., 2021). We will now survey three classes of synthetic tabular data synthesisers.

synthpop

The R package *synthpop* (Nowok, Raab and Dibben, 2016) was initially developed for synthesising tabular census data. *synthpop* models a joint distribution through a series of conditional distributions. The default option is to first model the marginal distribution of a selected starting variable, and then conditional distributions of the following variables, each conditioned on the preceding variables. This uses the following rule:

$$p\left(x_1, \ldots, x_n\right) = p\left(x_1\right) \cdot p\left(x_2 | x_1\right) \cdot \ldots \cdot p\left(x_n | x_1, \ldots, x_{n-1}\right).$$

The user can specify the order of the variables, which is named the visiting sequence, and there is also support for conditioning on only a specified set of predictor variables for each predicted variable, as long as all predictors are listed prior to the predicted variable in the visiting sequence. For example, if the visiting sequence is $x_1, x_2 \ldots, x_n$, we can use $p\left(x_3 | x_1\right)$ instead of $p\left(x_3 | x_1, x_2\right)$. *synthpop* supports many parametric and non-parametric synthesising methods for both continuous and discrete variables, which can be specified for each conditional distribution. Additionally, there is support for handling missing values and specifying rules for restricted values and dependent variables (Nowok, Raab and Dibben, 2016).

In summary, *synthpop* is a flexible package which is easy to implement and meets many of the challenges of generating synthetic tabular data. It handles a mixture of data types and includes correlations and dependencies between variables in the synthesising model. However, it may be limited for unbalanced and sparse data and other complex distributions.

Generative adversarial networks (GANs)

A generative adversarial network (GAN) is a deep learning model, and the interest in synthetic data escalated in the years after it was first proposed by

Goodfellow et al. (2014). While it may be that GANs are most known for synthetic image generation (Creswell et al., 2018), they are also frequently used for tabular data (Engelmann and Lessmann, 2021; Rajabi and Garibay, 2022; L. Xu, Skoularidou et al., 2019; Zhao et al., 2021).

This method consists of a generator and a discriminator, which in most applications are neural networks. The generator performs a mapping from a random noise signal to synthetic data, similar to our definition of a synthesiser in Definition 2.1.5. The discriminator is then given real and synthetic data samples, and returns predicted probabilities of the inputs being real. The two components have opposing objectives, since the discriminator aims to minimise the classification error, whereas the generator's objective is that the discriminator will fail to distinguish between the real and synthetic data. For each iteration, the parameters of the generator and discriminator are updated in turn. This iteration process guides the generator to provide synthetic data gradually closer to the distribution of real data, without directly seeing any real data, and the discriminator will detect increasingly subtle differences. The iteration process continues until convergence or when a predefined stopping criteria is reached. If we have converged to the optimal solution, the discriminator will predict 0.5 regardless of the entry being real or synthetic. This is the optimal stage of the generator, as the real and synthetic data have become indistinguishable (Creswell et al., 2018; Goodfellow et al., 2014).

Variants. There are many variants to the standard GAN, where some have different loss functions and others have an updated architecture. We will briefly introduce some of these, focusing on relevancy for tabular data.

The Conditional GAN (CGAN) (Mirza and Osindero, 2014) differs from the general GAN structure by having a condition c as an additional input to both the discriminator and generator. The condition can, for example, be a class, a specific value or an interval a variable is allowed to take. This makes it possible to generate synthetic data with specific properties, as the generator of the Conditional GAN (CGAN) returns synthesised data that follows condition c. The discriminator is given c along with real and synthetic data which follow condition c as an input.

The standard GAN and CGAN are not designed for tabular data, which makes them unfit for tabular data with variables following more complex distributions than the normal distribution, especially multi-modal distribution and long-tailed distributions. They are also not up to par for handling mixed data types and unbalanced categorical variables (L. Xu, Skoularidou et al., 2019). As a solution to this problem L. Xu, Skoularidou et al. (2019) developed the variant Conditional Tabular GAN (CTGAN), which supports data with mixed data types and multi-modal distributions. However, it lacked support for modelling missing values for continuous variables, and Zhao et al. (2021) made CTAB-GAN to solve this issue. CTAB-GAN also improved the synthesis of long-tailed distributions and sparse categorical distributions.

Challenges and disadvantages. An obstacle to using GANs is that it can be difficult to converge at an equilibrium state, which can often happen when the discriminator converges too quickly (Creswell et al., 2018). This is a variant of the vanishing gradient problem (Figueira and Vaz, 2022). Mode collapse

is another issue, which is when the synthetic data returned by the generator contains very similar observations, losing the multi-modality and heterogeneity of the real data (Creswell et al., 2018). This occurs because the generator has identified a few observations that the discriminator repeatedly classifies as real, making the generator focus on these observations instead of further exploring the data domain (Figueira and Vaz, 2022). This can be a great privacy challenge if the repeated data points are actual real data points. A general pitfall of neural networks is that they can overfit to the real data and memorise individual data points (Kuppa, Aouad and Le-Khac, 2021), and this also applies to GANs.

Other drawbacks relate to the fact that GANs are complex models. It can be hard to tune hyperparameters (Figueira and Vaz, 2022), and the computational cost is high. Additionally, since both the generator and discriminator are neural networks, the black box problem applies. There is no straight-forward way to tell what the GAN has learnt about the real data, contrary to *synthpop*, for which the joint distribution is explicitly defined.

2.2.2 Sequential Data

Sequential data differs from tabular data, as each observation consists of a data sequence, possibly of different lengths. The observations can also have constant covariates. This is a common data structure for health and sensor data, where measurements are taken at regular or irregular intervals (K. Zhang, Patki and Veeramachaneni, 2022). The goal is to generate synthetic sequential data where the dependence between the items in the sequence remains.

The Recurrent GAN (RGAN) and Recurrent Conditional GAN (RCGAN) were proposed by (Esteban, Hyland and Rätsch, 2018) and generate synthetic real-valued sequential data. They have the same frameworks as the GAN and CGAN, except that the discriminator and generator are both recurrent neural networks (RNNs). RNNs have a recursive structure, where the output is fed back into the network. This makes RNNs suitable for sequential data, since it can generate sequences where each element depends on previous elements (Bianchi et al., 2017).

Another synthesiser developed for sequential data is the Conditional Probabilistic Auto-Regressive (CPAR) model (K. Zhang, Patki and Veeramachaneni, 2022), where each element in a generated sequence also depends on the previous elements. CPAR has a single neural network structure and offers support for categorical and discrete data types. The model is available in the Python library *Synthetic Data Vault*.

2.2.3 Survival Data

Survival data or time-to-event data measures the time until an event occurs or is censored, and this event can also be correlated to a set of covariates. A proper introduction to survival analysis is given in Chapter 3. The field of synthetic survival data with censoring times is still in its early stages compared to tabular or sequential data. Recently, Norcliffe et al. (2023) presented the SurvivalGAN, which generates synthetic censored survival data. In short, SurvivalGAN consists of several synthesisers. First, it uses a tabular synthesiser to generate the covariates. Norcliffe et al. (2023) use Conditional Tabular GAN (CTGAN) for this purpose. Then, the covariates are entered into a survival

function fitted to the real data, which samples the time an event or censoring occurs. In another recent paper, Guillaudeux et al. (2023) present their Avatar method which is used to synthesise both survival and tabular patient data. Their method generates a synthetic avatar for each individual patient. The avatar is randomly drawn from the patient's local area, defined by its k nearest neighbours.

2.3 Privacy and Utility Trade-Off

In Definition 2.1.4 and the previous section, we alluded to that privacy and utility are two opposing goals and that we need to find a compromise between them. This is often referred to as a privacy and utility trade-off (Bellovin, Dutta and Reitinger, 2019; El Emam, 2020). Before we consider privacy and utility separately, we offer a short discussion of this trade-off.

We have two main objectives for the synthesiser $\mathcal{S}$. Firstly, the synthetic data $\mathcal{D}_s$ should hold the same properties as the real data $\mathcal{D}_r$, meaning that it is possible to get close to equal results from the two data sets in terms of inference, classification or other statistical or machine learning tasks. We refer to this aim as utility, which is a general term for the usefulness of the data (El Emam, 2020). Our second objective concerns privacy. The synthetic data should remain confidential and not disclose any real data points.

Drechsler (2011) bluntly states that it is impossible to generate a synthetic data set which is both completely private and provides high utility with respect to all potential use, meaning that a trade-off is unavoidable. By picturing two extremes, we can illustrate how the trade-off plays out. First, we want $\mathcal{D}_s$ to have full utility. As we recall from the discussion of the Synthesising Distribution Assumption (Raab, Nowok and Dibben, 2016) in Section 2.1.2, to obtain this we must generate $\mathcal{D}_s$ from the true model $\mathcal{T}$, so that $\mathcal{D}_r$ and $\mathcal{D}_s$ are samples from the same distribution. Unless the true model is previously known, we cannot find an $\mathcal{S}$ which is exactly equal to $\mathcal{T}$. Therefore, the only way to obtain $\mathcal{D}_s$ with full utility is to define $\mathcal{D}_s \coloneqq \mathcal{D}_r$. This will ensure that the two data sets share the exact same properties and structure, but will naturally breach every privacy concern. Oppositely, to ensure a fully private $\mathcal{D}_s$, it must be independent from $\mathcal{D}_r$. Then $\mathcal{D}_r$ cannot be used to fit $\mathcal{S}$, and consequently $\mathcal{D}_s$ will be unfit as a substitute for $\mathcal{D}_r$ (El Emam, 2020). Additionally, $\mathcal{D}_s$ no longer follows Definition 2.1.4, making it simulated and not synthetic data.

We can view this balancing act in a similar fashion as the bias-variance trade-off. A more complex $\mathcal{S}$ can provide higher utility, since we get a closer fit to the training data $\mathcal{D}_r$. If $\mathcal{S}$ is overfitted to $\mathcal{D}_r$, then $\mathcal{S}$ has high variance and low bias (Hastie, Tibshirani and Friedman, 2009). This can be at the expense of the data privacy of $\mathcal{D}_r$ (Bellovin, Dutta and Reitinger, 2019), because instead of learning the underlying data structure, $\mathcal{S}$ will generate $\mathcal{D}_s$ too close to $\mathcal{D}_r$. In the case where $\mathcal{S}$ is underfitted to $\mathcal{D}_r$ and has high bias and low variance, the model might not be complex enough to generate realistic $\mathcal{D}_s$ with high utility, but the privacy risk is low.

A delicate balance between privacy and utility must keep the sensitivity of the real data and the applications of the synthetic data in mind. In many situations, the privacy of the synthetic data cannot be compromised beneath a certain threshold. Thus, we need to find the synthetic data set with the

highest utility score with regards to this privacy threshold (Bellovin, Dutta and Reitinger, 2019; El Emam, 2020). If the search results in synthetic data with a too low utility score, it may be necessary to backtrack and attempt to develop a better synthesiser instead of releasing synthetic data with insufficient utility standards. In the remaining sections in this chapter, we will discuss specific ways of evaluating the privacy and utility, so that a meaningful tradeoff can be reached.

2.4 Utility Evaluation

We aim for synthetic data $\mathcal{D}_s$ that is indistinguishable from real data $\mathcal{D}_r$ in terms of utility, but how should this be measured? Beaulieu-Jones et al. (2019) ask domain experts to attempt to separate between $\mathcal{D}_r$ and $\mathcal{D}_s$ as a way of evaluating the utility. This is both time consuming and costly, and human domain experts may not be able to distinguish between the data as well as statistical and machine learning procedures, which is the standard approach.

The goal is to measure how well synthetic data performs as a substitute of real data, and we can do so by evaluating the performance of $\mathcal{D}_s$ compared to $\mathcal{D}_r$ on the intended tasks. It is more challenging to evaluate the utility of versatile synthetic data, which is required to perform well on many different tasks, most of which may not be known to us when we develop the synthesiser $\mathcal{S}$ and evaluate $\mathcal{D}_s$. Snoke et al. (2018) separate between *specific utility* and *general utility*, where the former evaluates how well $\mathcal{D}_s$ performs on specific statistical or machine learning tasks, and the latter evaluates the differences in distributions.

Figueira and Vaz (2022) and Kuppa, Aouad and Le-Khac (2021) use the terms *fidelity* and *diversity* instead of utility. Fidelity is closely linked to how we have defined utility so far, and is defined as a measure of how similar $\mathcal{D}_s$ is to $\mathcal{D}_r$ in terms of realism. Considering individual synthetic data points, could they be mistaken for real data points? The second term, diversity, refers to if $\mathcal{D}_s$ has the same variability and heterogeneity as $\mathcal{D}_r$. An $\mathcal{S}$ that only produces homogeneous data cannot have high utility, even if the synthetic data sets score high on fidelity. This is because $\mathcal{S}$ is unable to generate samples from the full domain space of $\mathcal{D}_r$. This is referred to as mode collapse in GAN literature (Creswell et al., 2018).

In this section we will present and discuss some commonly used evaluation methods for measuring the utility of synthetic data. We use utility as an umbrella term, and we will separate the evaluation methods into general and specific utility evaluation.

2.4.1 General Utility Evaluation

It is recommended to first start with general utility evaluation (Raab, Nowok and Dibben, 2021), as it is quicker to discover differences in distributions than performance on specific tasks. This is also typically done before any privacy specific evaluation. There exists many methods for this purpose, and while attempts have been made (Patki, Wedge and Veeramachaneni, 2016; Qian, Cebere and M. v. d. Schaar, 2023), there is no generally accepted framework. We will now introduce selected commonly used methods from the literature

which will be demonstrated in later chapters. These methods are primarily used for tabular data.

Plotting marginal distributions

For most data related tasks, it is good practice to begin by looking at the data. This also applies to evaluating the utility of synthetic data, and it is a common preliminary step (El Emam, 2020; Figueira and Vaz, 2022; Tucker et al., 2020). First, we check if the data format is the same for $\mathcal{D}_r$ and $\mathcal{D}_s$, which means inspecting if the variables have the same data types and that there are no illegal values. Next, we plot the marginal probability density functions (PDFs) of continuous variables for both $\mathcal{D}_r$ and $\mathcal{D}_s$ and visually determine if they are similar or not. The probability mass function of finite discrete variables can be compared through frequency tables or histograms. If the marginal distributions of $\mathcal{D}_r$ and $\mathcal{D}_s$ look very different, we should adjust $\mathcal{S}$ rather than evaluating $\mathcal{D}_s$ further.

Kolmogorov-Smirnov (KS) test

It is not sufficient to state that two marginal distributions look the same, and we need a specific method to measure the distance between the distributions. For continuous variables we can use the two-sided Kolmogorov-Smirnov (KS) test, which is commonly used for utility evaluation of synthetic data (El Emam, 2020; Tucker et al., 2020).

The KS test compares continuous distributions by considering the maximum absolute distance between their cumulative graphs (Kauermann, Küchenhoff and Heumann, 2021). The null hypothesis H_0 is that the real and synthetic samples are drawn from the same distribution (R Core Team, 2023).

Keep in mind that the test can reveal a significant difference even for samples with small differences when the number of observations is large (Tucker et al., 2020). If the distances are too close, this can potentially indicate overfitting, so a small distance might be preferable. We will expand on this idea in Section 2.6.

Chi-square goodness-of-fit test

For categorical and discrete variables, a chi-square goodness-of-fit test is often used to evaluate the difference in expected frequencies between real and synthetic marginal distributions (El Emam, 2020; Steinbakk, Langsrud and Løland, 2020; Tucker et al., 2020).

Definition 2.4.1 [Chi-square goodness-of-fit test]

Say we have a sample of synthetic values $X_1, \ldots, X_n$ and a sample of real values $Y_1, \ldots, Y_n$, which can take K categorical values. The counts of each value are $i_k = \sum_{l=1}^{n} \mathbf{1}_{X_l=k}$ for the synthetic values and $j_k = \sum_{l=1}^{n} \mathbf{1}_{Y_l=k}$ for the real values. $P_X(X = k) = p_k$ are the probabilities of the synthetic values, and we want to test if they are equal to the frequencies of $\mathcal{D}_r$, $\hat{P}_k = j_k/n$. Then, we have the following hypotheses:

$$H_0 : p_1 = \frac{j_1}{n}, \ldots, p_K = \frac{j_K}{n}$$
$$H_a : H_0 \text{ is not true.}$$

The test statistic is

$$\chi^2 = \sum_{k=1}^{K} \frac{(i_k - j_k)^2}{j_k}.$$

Under H_0, χ^2 approximately follows a chi-squared distribution with $K - 1$ degrees of freedom. A α level test rejects H_0 if

$$\chi^2 \geq \chi^2_{\alpha, K-1}$$

(Devore, Berk and Carlton, 2021; Kauermann, Küchenhoff and Heumann, 2021).

As for the KS test, we must remember that the chi-square goodness-of-fit test can reject H_0 when the differences are small if n is large.

Correlation

Similar marginal distributions are of little value if the correlation structure is not maintained. This is emphasised in Definition 2.1.3. In order to control that joint distributions are similar, we can compare pairwise correlation matrices of $\mathcal{D}_r$ and $\mathcal{D}_s$. This is a commonly used method for evaluating synthetic data. Both Pearson and Spearman correlation are used (Beaulieu-Jones et al., 2019; Figueira and Vaz, 2022; Zhao et al., 2021).

The mean difference in pairwise correlations is used as an evaluation metric (Kokosi et al., 2022; Steinbakk, Langsrud and Løland, 2020). This also allows for assigning more weight to specified correlations, which can be useful in cases where it is important that the correlations between certain pairs of variables are correctly represented in $\mathcal{D}_s$.

2.4.2 Specific Utility Evaluation

A separation between specific and general utility can be misleading, because the specific utility evaluation methods also aim to provide a general evaluation. This is because we aim for creating synthetic data that can be used for a wide range of tasks. If we know what kinds of tasks the synthetic data is intended for, we will evaluate based on them. However, we need a general mindset, because these tasks may not be known. Specific utility evaluation compares the performance of the real and synthetic data on a broad range of specific tasks, with the aim of measuring how well the synthetic data generalise to new unattempted tasks.

Compare point estimates and confidence intervals

Easing into specific utility tasks, we begin with a comparison of point estimates and confidence intervals of statistics from $\mathcal{D}_r$ and $\mathcal{D}_s$ (Snoke et al., 2018). We

can for example use summary statistics of the marginal distributions, such as their estimated means, standard errors and selected percentiles (Figueira and Vaz, 2022; Steinbakk, Langsrud and Løland, 2020). Parameters of simple parametric models such as coefficients of regression models (Snoke et al., 2018) or hazard ratio coefficients for survival data (Guillaudeux et al., 2023) can also be useful. While this is similar to the general utility evaluation, Snoke et al. (2018) place this in the specific utility evaluation category as it is an inference task, relating to the use cases discussed in Section 2.1.3. If $\mathcal{D}_r$ and $\mathcal{D}_s$ have similar point estimates, this indicates good utility.

As an overall metric, Snoke et al. (2018) recommends the standardised difference utility metric:

Definition 2.4.2 [Standardised difference utility metric]

For a statistic η, where the estimate from $\mathcal{D}_r$ and $\mathcal{D}_s$ are $\hat{\eta}_s$ and $\hat{\eta}_r$ respectively, with standard errors $se\left(\hat{\eta}_s\right)$ and $se\left(\hat{\eta}_r\right)$. the standardised difference utility metric is

$$SDU = \frac{|\hat{\eta}_r - \hat{\eta}_s|}{se\left(\hat{\eta}_r\right)}.$$

For confidence intervals, we can use the interval overlap utility metric, which is defined below.

Definition 2.4.3 [Interval overlap utility metric]

Given any statistic, the estimate from $\mathcal{D}_s$ has the confidence interval $[L_s, U_s]$ and the estimate from $\mathcal{D}_r$ has the confidence interval $[L_r, U_r]$. Then the interval overlap metric is

$$IOU = \frac{\min\left(U_s, U_r\right) - \max\left(L_s, L_r\right)}{2\left(U_r - L_r\right)} + \frac{\min\left(U_s, U_r\right) - \max\left(L_s, L_r\right)}{2\left(U_s - L_s\right)}$$

(Drechsler, 2011; Snoke et al., 2018).

When the intervals fully overlap, IOU is equal to 1, which is the best utility score. When there is no overlap, the metric is negative. Both SDU and IOU are implemented in *synthpop* (Nowok, Raab and Dibben, 2016).

Machine learning efficacy

It is common to use machine learning tasks for specific utility evaluation, and particularly supervised learning tasks. They are suitable for utility evaluation, because the difference in performance between $\mathcal{D}_r$ and $\mathcal{D}_s$ for a specific model is easy to interpret. Machine learning tasks are also a common use case for synthetic data, as discussed in Section 2.1.3.

To the best of our knowledge, Esteban, Hyland and Rätsch (2018) were first to propose the method "Train on synthetic, test on real" (TSTR), which has since been widely adopted under various names (Breugel, Qian and M. v. d.

Schaar, 2023; Jordon, Yoon and M. v. d. Schaar, 2018; Norcliffe et al., 2023; Tucker et al., 2020; Xie et al., 2018). The method assesses how well models fitted to synthetic data perform on unseen real data. Zhao et al. (2021) pair them with another method that we for continuity will name "Train on real, test on real" (TRTR), and Norcliffe et al. (2023) present a variant which they call "Train on synthetic, test on synthetic" (TSTS).

Algorithm 1 provide TSTR and TRTR jointly. It is closely based on Algorithm 1 in Esteban, Hyland and Rätsch (2018), but with some differences. In the original algorithm the model $\mathcal{SL}$ is a classifier with a specified target variable, and it does not contain line 4 and 6, as this relates to TRTR. In line 1, we separate the real data into a training and test set. It is important that the data $\mathcal{D}_r^{(test)}$ used to evaluate the model fitted to the synthetic data, $\mathcal{SL}_s$, is completely separate from the data used to train the synthesiser, $\mathcal{D}_r^{(train)}$. Despite that $\mathcal{D}_r^{(train)}$ is not used to train $\mathcal{SL}_s$, it can still influence the model through $\mathcal{D}_s$. In line 2 we generate $\mathcal{D}_s$ using the synthesiser $\mathcal{S}$. Next, in lines $3-4$, we fit an arbitrary supervised learning model to $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$ respectively. Lastly, in lines $5-6$, we find the performance of the fitted models on the separate real data set $\mathcal{D}_r^{(test)}$.

Algorithm 1 "Train on synthetic, test on real" (TSTR) and "train on real, test on real" (TRTR)

Input:
$\mathcal{D}_r \leftarrow$ real data
$\mathcal{S} \leftarrow$ synthesiser
$\mathcal{SL} \leftarrow$ supervised learning model
Output:
$\text{eval}_s \leftarrow$ performance score for $\mathcal{D}_s$
$\text{eval}_r \leftarrow$ performance score for $\mathcal{D}_r$

1: $\mathcal{D}_r^{(train)}, \mathcal{D}_r^{(test)} \leftarrow \text{split}\left(\mathcal{D}_r\right)$
2: $\mathcal{D}_s \leftarrow \mathcal{S}\left(\mathcal{D}_r^{(train)}\right)$
3: $\mathcal{SL}_s \leftarrow \text{train}\left(\mathcal{SL}, \mathcal{D}_s\right)$
4: $\mathcal{SL}_r \leftarrow \text{train}\left(\mathcal{SL}, \mathcal{D}_r^{(train)}\right)$
5: $\text{eval}_s \leftarrow \text{score}\left(\mathcal{D}_r^{(test)}, \mathcal{SL}_s\right)$
6: $\text{eval}_r \leftarrow \text{score}\left(\mathcal{D}_r^{(test)}, \mathcal{SL}_r\right)$

Classifiers are a popular choice of model, and is paired with the area under the receiver operating characteristic (AUROC) curve to evaluate the performance (Breugel, Qian and M. v. d. Schaar, 2023; Guillaudeux et al., 2023; Tucker et al., 2020; Zhao et al., 2021), but in principle any supervised learning model with a suitable scoring function can be used.

The choice of appropriate models for utility evaluation also depends on the data type. For survival data, we can test if $\mathcal{D}_r$ and $\mathcal{D}_s$ come from the same underlying distribution by comparing the Kaplan-Meier survival curves (Norcliffe et al., 2023) using a log-rank test (Guillaudeux et al., 2023; Kleinbaum and Klein, 2005). Survival curves are formally introduced in Chapter 3. Moreover,

models that allow for a variable importance ranking, such as random forests, can be used to check if $\mathcal{D}_r$ and $\mathcal{D}_s$ provide a similar ranking (Beaulieu-Jones et al., 2019; Fan, 2020).

Norcliffe et al. (2023) use TSTS to evaluate the utility of $\mathcal{D}_s$ for model selection, which we have discussed in Section 2.1.3. For a selection of models, we can both train and test them on separate sets from $\mathcal{D}_r$ and $\mathcal{D}_s$. If the models trained and tested on $\mathcal{D}_s$ are ranked equally in terms of performance as the same models trained and tested on $\mathcal{D}_r$, this indicates that $\mathcal{D}_s$ can be used for model selection.

2.5 Privacy Evaluation

When Rubin (1993) first proposed synthetic data, he argued that since no individual's data would ever be released, this would automatically meet any privacy requirements. Similarly, Duncan, Elliot and Salazar-González (2011) stated that synthetic data makes identity disclosure almost impossible in most cases. This turned out to be too optimistic. Despite being considered safer than anonymisation and augmentation methods, synthetic data also poses a risk of identity disclosure (Stadler, Oprisanu and Troncoso, 2022). Since we learn $\mathcal{S}$ from $\mathcal{D}_r$, it is inevitable that some information about individual points in $\mathcal{D}_r$ leaks into $\mathcal{D}_s$. Consequently, is it important that this information leakage is controlled and limited.

2.5.1 Types of Privacy Disclosure

Identity disclosure is one of three common categories of privacy disclosure, together with attribute disclosure and inferential disclosure (El Emam, 2020; Shlomo, 2018). In this section we define each of these terms and discuss how they relate to the privacy of synthetic data and adversarial attacks.

Identity disclosure occurs when we can identify individuals from released data sets. This is usually discussed in relation to pseudonymised data sets and linkage attacks (Shlomo, 2018), where an observation is linked to a specific identity using prior knowledge (El Emam, 2020). Most synthesisers do not map individual points from the real data to a specific synthetic point. Instead, the synthetic data points are generated based on knowledge of a set of real points. Then, it is not possible to match a synthetic point to a single real identity. However, identity disclosure can take place based on the *whole* synthetic data set. We will explore this further in Section 2.5.2.

Attribute disclosure is a case where an adversary gains information about an individual without being able to correctly assign their identity to a specific observation. Say that an adversary knows that an individual is in the data set, and has access to enough of its attribute values to assign the individual to a group. If every member of the group has the same value for some attribute, then the adversary knows that this is also the case for the targeted individual (El Emam, 2020; Shlomo, 2018). This can also be an issue for synthetic data sets. If the synthetic data contains groups with (close to) no variation with respect to some attribute, then this is likely the case in the real training data as well. Therefore, it is important that the synthesising process allows for added noise. Smaller groups and outliers require more noise than larger groups.

Inferential disclosure is a related issue, where group membership can cause high confidence in that an individual has a specific attribute value, even when this individual is not in the real training data of the synthesiser (Shlomo, 2018). El Emam (2020) argues that since this information gain is not absolutely certain, this should not be an issue. Further, he states that by not allowing for inferential disclosure, we will in practice not be able to use the synthetic data for any type of data analysis task, defeating the point of synthetic data altogether.

A general trend is that using larger real data set to fit the synthesiser leads to less privacy exposure of individual observations, since the synthesiser has access to more data material and single observations have less weight. This also makes it less likely for groups to have exactly the same values for certain attributes. However, outliers can still be at risk and should be accounted for during the privacy evaluation process.

2.5.2 Membership Inference Attacks

Membership inference attacks (MIAs) are an increasing concern today. In these attacks, an adversary uses machine learning models to identify individual points, members, in an unknown training data set. They do so by targeting a known model fitted to the training data (Chen, Yu et al., 2020; Shokri et al., 2017). The attack utilises weaknesses in overfitted models that contain information about specific members in the training data, and attempts to reveal this information. MIAs are also a concern for synthetic data. Even when the synthesiser $\mathcal{S}$ is not released, an adversary can conduct an MIA by using the output $\mathcal{D}_s$ to predict if a data point is present in the real training data $\mathcal{D}_r$ or not.

Kuppa, Aouad and Le-Khac (2021) present a method for performing such synthetic data attacks, building on ideas of model attacks proposed by Shokri et al. (2017). Their method requires that the adversary has access to a data set similar to $\mathcal{D}_r$, which we name $\mathcal{B}$. The goal is to use $\mathcal{B}$ to train a classifier that given a synthetic data set $\mathcal{D}_s$ and a point x can predict if $x \in \mathcal{D}_r$, i.e. if x is a member of the training data set used to synthesise $\mathcal{D}_s$. We say that x is a suspected member of $\mathcal{D}_r$ and the candidate of the attack. The classifier is trained by using bagging to obtain many labelled synthetic data sets.

Shokri et al. (2017) refers to such a classifier as an attack model, and Algorithm 2 describes how we can train it. If $x \in \mathcal{B}$, we will first remove it (line 1-2). By bootstrap sampling from $\mathcal{B}$, we obtain $B_1, \ldots, B_k$ of size $n \leftarrow |\mathcal{B}|$ in line 6. We call them shadow data sets. The synthesiser $\mathcal{S}$ is unknown, but we use the shadow data sets to fit imitations that we call shadow synthesisers. Information about the model structure, for example that $\mathcal{S}$ is a GAN, might be known to an adversary. However, if no such information is available, the adversary needs to make an educated guess on the model structure. For each $B_1, \ldots, B_k$ we fit synthesisers of the chosen model structure and get the shadow synthesisers $S_1, \ldots, S_k$, which generate the synthetic data sets $D_{s,1} \ldots D_{s,k}$ respectively, all of size n (line 7-8). In a similar for loop (line 10-15), this is repeated, but now each bootstrap sample will be of size $n - 1$, and then we add x to each $B'_1, \ldots, B'_k$ (line 12). We then fit new shadow synthesisers and obtain the synthetic data sets $D'_{s,1} \ldots, D'_{s,k}$. While the synthetic data sets $\boldsymbol{D}_s = \{D_{s,1}, \ldots, D_{s,k}\}$ do not have the candidate data point x in the training

sets of their shadow synthesisers, this is the case for $\boldsymbol{D}'_s = \left\{D'_{s,1}, \ldots, D'_{s,k}\right\}$. Therefore, sensitive information about x may be present in the synthetic data sets in $\boldsymbol{D}'_s$. We attach the label 0 to each set in $\boldsymbol{D}_s$ and 1 to each set in $\boldsymbol{D}'_s$, and fit a synthesiser that given a synthetic data set will predict 0 or 1.

Algorithm 2 Preparing an MIA on synthetic data

Input:
$\mathcal{B} \leftarrow$ data set similar to $\mathcal{D}_r$
$k \leftarrow$ amount of synthetic data set pairs
$x \leftarrow$ candidate data point
Output:
$\mathcal{C}_x \leftarrow$ classifier which given $\mathcal{D}_s$ can predict if $x \in \mathcal{D}_r$

1: **if** $x \in \mathcal{B}$ **then**
2: $\quad \mathcal{B} \leftarrow \mathcal{B} \setminus \{x\}$
3: **end if**
4: $n \leftarrow |\mathcal{B}|$
5: **for** $i \in \{1, \ldots, k\}$ **do**
6: $\quad B_i \leftarrow$ Bootstrap sample from $\mathcal{B}$ of size n
7: $\quad$ Fit S_i to B_i
8: $\quad D_{s,i} \leftarrow n$ synthetic data points generated from S_i
9: **end for**
10: **for** $i \in \{1, \ldots, k\}$ **do**
11: $\quad B'_i \leftarrow$ Bootstrap sample from $\mathcal{B}$ of size $n-1$
12: $\quad$ Add x to B'_i
13: $\quad$ Fit S'_i to B'_i
14: $\quad D'_{s,i} \leftarrow n$ synthetic data points generated from S'_i
15: **end for**
16: Use $\boldsymbol{D}_s$ and $\boldsymbol{D}'_s$ to train a classifier $\mathcal{C}_x$

For each candidate data point x, an adversary can train a classifier $\mathcal{C}_x$, and use it to predict if x is a member of the training set used to fit the synthesiser $\mathcal{S}$ that $\mathcal{D}_s$ is generated from. If $\mathcal{C}_x$ returns 1 it predicts that $x \in \mathcal{D}_r$, and 0 predicts that $x \notin \mathcal{D}_r$. The classifier may also return the confidence score of this prediction.

In many cases, the adversary will not have a specific suspected data point in mind. **We propose a novel form of attack**, where $\mathcal{C}_x(\mathcal{D}_s)$ is treated as a function of x, and then we search through the domain $\mathcal{X}$ to find values $x \in \mathcal{X}$ for which $\mathcal{C}_x(\mathcal{D}_s)$ returns a high score. This assumes that $\mathcal{C}_x$ returns the confidence score of its prediction. A possible search algorithm is hill-climbing, which is used in Algorithm 3. We begin by a randomly selecting a point $x \in \mathcal{X}$ in line 1. Next, we use Algorithm 2 to fit a classifier in line 2, which gives us a confidence score $S = \mathcal{C}_x(\mathcal{D}_s)$ in line $2-3$. We then consider all neighbouring points of x, fit corresponding classifiers (line 6), and select the neighbour with the highest score (line 11. The search terminates when we find a local optimum, meaning that there are no neighbouring points with higher scores. By repeating Algorithm 3 multiple times, we can find many candidates for data points in $\mathcal{D}_r$.

Algorithm 3 can also be used for attribute disclosure. If an adversary knows that a specific member was present in the training data, but only knows parts

Algorithm 3 Hill-climbing in an MIA

Input:
$\mathcal{B} \leftarrow$ data set similar to $\mathcal{D}_r$
$k \leftarrow$ amount of synthetic data set pairs
$\mathcal{D}_s \leftarrow$ synthetic data set
Output:
$x* \leftarrow$ local maximum
Randomly select $x*$ from $\mathcal{X}$
$\mathcal{C}_x* \leftarrow$ Algorithm 2$(\mathcal{B}, k, x*)$
$S = \mathcal{C}_{x*}(\mathcal{D}_s)$
while S is unconverged **do**
  for each $x'_i \in$ neighbour(x) **do**
    $S_i = \mathcal{C}_{x'_i}(\mathcal{D}_s)$
  end for
  $S* = \max\{S_1, \ldots, \S_i, \ldots\}$
  if $S* > S$ **then**
    $S = S*$
    $x* \leftarrow x$-value corresponding to S
  end if
end while

of the attributes of the member, they can do a search conditioned on the known attributes. The returned optimum will be a data point with the known attributes values and highly probable values for the unknown attribute values.

The idea of using hill-climbing to identify highly probable members is closely inspired by Shokri et al. (2017), which performs an attack on the training set of a classifier. Their idea is built on the assumption that probabilistic classifier returns higher probabilities of members in its training set than unseen data. Thus, the classifier can be used to generate the data set $\mathcal{B}$ when real data similar to the training data is not accessible to the adversary. They do this by searching through the domain space using hill-climbing like above, and collect local optimums in $\mathcal{B}$. Our idea differs, as we perform hill-climbing using the attack model instead of a targeted classifier. Moreover, we wish to find data points with a high probability of being members of the training data, whilst Shokri et al. (2017) aim to generate the data set $\mathcal{B}$ used for the attack. Due to time and computational constraints, this novel type of attack is not included in our MIA experiment in section 4.6.

There are several weaknesses to an MIA on synthetic data. Firstly, in many cases it will be hard for an adversary to get access to a real data set $\mathcal{B}$ that it is close to $\mathcal{D}_r$. In the case $\mathcal{B}$ is available, the adversary cannot know how close it is to $\mathcal{D}_r$, and by that know how accurate $\mathcal{C}_x$ is. The solution by Shokri et al. (2017) sketched out above of using the target model to generate $\mathcal{B}$ will not work in the same way, as the target model $\mathcal{S}$ is not a classifier. In our case it is more straightforward to use $\mathcal{S}$ to generate a data set, which opens for the possibility of using another synthetic data set in place of $\mathcal{B}$. Secondly, if the adversary in addition needs to guess the structure of $\mathcal{S}$, the results will be more dubious. However, successful MIAs have been made even when the attacked model is a complete black box and the available data set $\mathcal{B}$ is not very similar to $\mathcal{D}_r$ (Shokri

et al., 2017). Thirdly, Algorithm 3 has the potential of being computationally demanding, since it requires that many synthesisers are trained. For larger data sets it can also be a challenge to check how sensitive each point in $\mathcal{D}_r$ is to MIAs. On the contrary, increasing the size of $\mathcal{D}_r$ decreases the accuracy of an attack (Fan, 2020; Lin, Sekar and Fanti, 2021; C. Xu et al., 2019).

In this section we have discussed attacks that utilises how two synthesisers fitted to data sets that differ by a specific observation are different, and how these differences can be discovered from their generated synthetic data sets. These attacks allow for a measurement of the identity disclosure risk of each member in $\mathcal{D}_r$, but they are no guarantee against future successful attacks. In the next section we will discuss a method that formally limits the identity disclosure risk of every point in $\mathcal{D}_r$.

2.5.3 Differential Privacy

Differential privacy is a property of a mechanism that offers a formal privacy guarantee. Because of this, there is great interest in differentially private synthesisers (Beaulieu-Jones et al., 2019; Bonomi, Jiang and Ohno-Machado, 2020; Bowen and Snoke, 2021; Chen, Cheung et al., 2021; Esteban, Hyland and Rätsch, 2018; Jordon, Yoon and M. v. d. Schaar, 2018; Ping, Stoyanovich and Howe, 2017; Xie et al., 2018). By making a synthesiser $\mathcal{S}$ differentially private, the privacy protection task is incorporated into the synthesis process, which we mentioned in Section 2.1.2. Therefore, every synthetic data set $\mathcal{D}_s$ generated by $\mathcal{S}$ will uphold a given level of privacy for all points in the real training data $\mathcal{D}_r$. In this section we present the basic definitions and concepts of differential privacy. We will mostly follow the notation of Dwork and Roth (2014) and Dwork and Smith (2010), which use mechanism as a general term for any algorithm which takes a data set as input and returns an arbitrary output. We first consider differential privacy for a general mechanism, and then examine differential privacy through a synthetic data lens.

Differential privacy determines how much a single data point can contribute to the output of a mechanism. Say that $\mathcal{M} : \mathbb{R}^{n\times m} \to O$ is an arbitrary randomised mechanism, and that $\mathcal{D} \in \mathbb{R}^{n\times m}$ and $\mathcal{D}' \in \mathbb{R}^{n\times m}$ are two data sets that differ by a single row. This means that for $D = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $D' = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$, we have that $\boldsymbol{x}_i = \boldsymbol{y}_i$ for all $i \neq j$, where j is a single index in $\{1, \ldots, n\}$. If $\mathcal{M}$ is differentially private then the probability of obtaining any output in a subspace $S \subseteq O$ should be close to the same when we use $\mathcal{D}$ or $\mathcal{D}'$ as input. Then, there is a limit to the amount of information about a single observation that can be revealed from the output of $\mathcal{M}$.

Definition 2.5.1 [ε-differential privacy]

A randomised mechanism $\mathcal{M} : \mathbb{R}^{n\times m} \to O$ is ε-differentially private if for all data sets $D, D' \in \mathbb{R}^{n\times m}$ differing by at most one row, and for all $S \subseteq O$, where O is an arbitrary output space, we have that

$$P\left[\mathcal{M}(D) \in S\right] \leq \exp(\varepsilon)\, P\left[\mathcal{M}\left(D'\right) \in S\right],$$

where $\varepsilon \geq 0$ (Dwork and Roth, 2014, Definition 2.4).

This definition can be difficult to interpret, but if the inequality is rewritten into

$$\frac{P\left[\mathcal{M}\left(D\right) \in S\right]}{P\left[\mathcal{M}\left(D'\right) \in S\right]} \leq \exp\left(\varepsilon\right), \tag{2.1}$$

this motivates an alternative interpretation of ε. The left hand side of the inequality (2.1) is the ratio between the probabilities that $\mathcal{M}$ will return a value in S conditioned on the inputs D and D' respectively. If the probabilities are the same and $\varepsilon = 0$, then all $\boldsymbol{x} \in D$ and $\boldsymbol{y} \in D'$ have full privacy, and the output space S is independent of both $\boldsymbol{x}$ and $\boldsymbol{y}$ (Dwork and Roth, 2014). However, this means that $\mathcal{M}$ cannot learn anything from neither D nor D', and we have again encountered a privacy and utility tradeoff. Increasing values of ε allow for a larger difference between the probabilities, and thus a larger dependence between a single point $\boldsymbol{x} \in D$ and the output. This increases utility and decreases privacy.

We often refer to ε as the privacy budget of $\mathcal{M}$. Each time we get an output of an $\varepsilon-$differentially private mechanism, we use a privacy budget of ε. This makes intuitively sense, since queries about a data set that in itself poses a small privacy treat can grow to become a great concern if we aggregate information by many queries. If we need to keep the same level of privacy, the amount of noise must increase with the number of queries. Dwork and Roth (2014) call this the query release problem. This relates to the composition property of differential privacy, which states that a mechanism consisting of n $\varepsilon-$differentially private mechanisms is $n\varepsilon-$differentially private (Dwork and Roth, 2014). The composition property allows us to control the combined privacy budget of multiple mechanisms.

Another important property is the post-processing property. It states that no post-processing step performed on the output of an $\varepsilon-$differentially private mechanism can reduce the guarantee (Dwork and Roth, 2014, Proposition 2.1). This means that no matter the complexity of adversarial attacks, ε regulates how much information it is possible to collect from a single output of an $\varepsilon-$differentially private mechanism.

The Laplace mechanism

Definition 2.5.1 requires that the mechanism must be randomised in order to be differentially private. This is essential, because randomised noise complicates a potential mapping between input and output. The amount of noise should be proportional to how much a single data point can contribute to the outcome. One way to measure this contribution is through l_1 sensitivity, which states the maximum contribution of one data point to the outcome of a function.

Definition 2.5.2 [l_1 sensitivity]

Given a function $f : \mathbb{R}^{n\times m} \to O$, for any $D, D' \in \mathbb{R}^{n\times m}$ that differ by at most one row, the l_1 sensitivity of f is

$$\Delta f = \max_{D,D'\in\mathbb{R}^{n\times m}} \|f\left(D\right) - f\left(D'\right)\|_1,$$

where $\|\|\|_1$ is the L_1-distance (Dwork and Roth, 2014, Definition 3.1).

By adding randomised noise of the same magnitude as the l_1 sensitivity to the output, we can obtain differential privacy. This is called the Laplace mechanism (Dwork and Roth, 2014; Dwork and Smith, 2010).

Definition 2.5.3 [The Laplace mechanism]

A function $f : \mathbb{R}^{n\times m} \to \mathbb{R}^l$ has l_1 sensitivity Δf. The Laplace mechanism $\mathcal{M}_L$ satisfies ε-differential privacy and is defined as

$$\mathcal{M}_L\left(D, f, \varepsilon\right) = f\left(D\right) + \left(Y_1, \ldots, Y_l\right),$$

where $Y \overset{\text{iid}}{\sim} \text{Lap}\left(\Delta f/\varepsilon\right)$. (Dwork and Roth, 2014, Definition 3.3; Dwork and Smith, 2010, Theorem 6).

We include the proof presented by Dwork and Roth (2014), because it utilises techniques that will be of later use.

Proof. Following Definition 2.5.1, our task is to show that for any $\boldsymbol{z} \in \mathbb{R}^l$ we have that

$$\frac{P\left(\mathcal{M}_L\left(D, f, \varepsilon\right) = \boldsymbol{z}\right)}{P\left(\mathcal{M}_L\left(D', f, \varepsilon\right) = \boldsymbol{z}\right)} \leq \exp\left(\varepsilon\right) \tag{2.2}$$

for data sets $D, D' \in \mathbb{R}^{n\times m}$ differing by one row.
If $Y \overset{\text{iid}}{\sim} \text{Lap}\left(\Delta f/\varepsilon\right)$, then its PDF is

$$p_Y\left(y|\Delta f/\varepsilon\right) = \frac{\varepsilon}{2\Delta f}\exp\left(-\frac{\varepsilon|x|}{\Delta f}\right).$$

By using that $Y \overset{\text{iid}}{\sim} \text{Lap}\left(\Delta f/\varepsilon\right)$, we show that

$$\begin{aligned} P\left(\mathcal{M}_L\left(D, f, \varepsilon\right) = \boldsymbol{z}\right) &= P\left(f\left(D\right) + \left(Y_1, \ldots, Y_l\right) = \boldsymbol{z}\right) \\ &= \prod_{i=1}^{l} p_Y\left(z_i - f\left(D\right)_i|\Delta f/\varepsilon\right). \end{aligned}$$

We can then write the left hand side of inequality (2.2) as

$$
\begin{aligned}
\frac{P\left(\mathcal{M}_L\left(D, f, \varepsilon\right) = \boldsymbol{z}\right)}{P\left(\mathcal{M}_L\left(D', f, \varepsilon\right) = \boldsymbol{z}\right)} &= \prod_{i=1}^{l} \frac{p_Y\left(z_i - f\left(D\right)_i | \Delta f / \varepsilon\right)}{p_Y\left(z_i - f\left(D'\right)_i | \Delta f / \varepsilon\right)} \\
&= \prod_{i=1}^{l} \frac{\exp\left(\left(-\varepsilon / \Delta f\right) | z_i - f\left(D\right)_i |\right)}{\exp\left(\left(-\varepsilon / \Delta f\right) | z_i - f\left(D'\right)_i |\right)} \\
&= \prod_{i=1}^{l} \exp\left(\frac{\varepsilon}{\Delta f}\left(|z_i - f\left(D'\right)_i| - |z_i - f\left(D\right)_i|\right)\right) \\
&\leq \prod_{i=1}^{l} \exp\left(\frac{\varepsilon}{\Delta f}\left(|z_i - f\left(D'\right)_i - z_i + f\left(D\right)_i|\right)\right) \\
&= \exp\left(\frac{\varepsilon}{\Delta f}\left(\sum_{i=1}^{l} |f\left(D\right)_i - f\left(D'\right)_i|\right)\right) \\
&= \exp\left(\frac{\varepsilon}{\Delta f} \Delta f\right) = \exp\left(\varepsilon\right).
\end{aligned}
$$

The inequality follows from the triangle inequality, and

$$
\sum_{i=1}^{l} |f\left(D'\right)_i - f\left(D\right)_i| = \Delta f
$$

follows from Definition 2.5.2. ■

The exponential mechanism

The l_1 sensitivity finds the maximum possible difference between any two neighbouring data sets, which means that potential outliers have a great influence on the sensitivity. Most neighbouring data sets are closer than the l_1 sensitivity, but in order to protect any $\boldsymbol{x} \in \mathbb{R}^m$ equally, the Laplace mechanism takes the worst case scenario into account. As a consequence, there is a risk that the amount of added noise will conceal any meaningful signal to the output. This balance between signal and noise is another case of the privacy-utility trade-off. We wish to secure the training data by adding noise to the output, but in such a way that the output of the differentially private mechanism is still useful. The exponential mechanism (Dwork and Roth, 2014) offers a way of controlling the noise signal's influence on the utility, by making an output with high utility more probable. For a given ε, we need to find the optimal $\varepsilon-$differentially private mechanism in terms of utility. To do so, we need a utility function $u\left(D, \boldsymbol{z}\right)$, which maps the input and output of the mechanism to a utility score (Dwork and Roth, 2014).

Definition 2.5.4 [The exponential mechanism]

An exponential mechanism $\mathcal{M}_E\left(u, D\right)$ takes input data $D \in \mathbb{R}^{n \times m}$, and a

utility function $u(D, \boldsymbol{z})$ with l_1 sensitivity Δu. If

$$P\left(\mathcal{M}_E(u, D) = \boldsymbol{z}\right) \propto \exp\left(\frac{\varepsilon \cdot u(D, \boldsymbol{z})}{2\Delta u}\right),$$

for any value $\boldsymbol{z} \in \mathcal{Z}$, then $\mathcal{M}_E$ is $\varepsilon-$deferentially private (Dwork and Roth, 2014; Nguyên and Hui, 2018).

We include the proof from Dwork and Roth (2014), with some adjustments. In our version of the proof, $\mathcal{Z}$ is a continuous, and not discrete, space.

Proof. For two neighbouring data sets D and D', we have that for any $\boldsymbol{z} \in \mathcal{Z}$

$$\begin{aligned}
&\frac{P\left(\mathcal{M}_E(U, D) = \boldsymbol{z}\right)}{P\left(\mathcal{M}_E(U, D') = \boldsymbol{z}\right)} \\
=&\frac{\left(\exp\left(\varepsilon \cdot u(D, \boldsymbol{z})/2\Delta u\right)\right) / \left(\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u\left(D, \boldsymbol{z}'\right)/2\Delta u\right) d\boldsymbol{z}'\right)}{\left(\exp\left(\varepsilon \cdot u(D', \boldsymbol{z})/2\Delta u\right)\right) / \left(\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u(D', \boldsymbol{z}')/2\Delta u\right) d\boldsymbol{z}'\right)} \\
=&\frac{\exp\left(\varepsilon \cdot u(D, \boldsymbol{z})/2\Delta u\right)}{\exp\left(\varepsilon \cdot u(D', \boldsymbol{z})/2\Delta u\right)} \cdot \frac{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u\left(D', \boldsymbol{z}'\right)/2\Delta u\right) d\boldsymbol{z}'}{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u(D, \boldsymbol{z}')/2\Delta u\right) d\boldsymbol{z}'} \\
\leq&\exp\left(\frac{\varepsilon\left(u(D, \boldsymbol{z}) - u\left(D', \boldsymbol{z}\right)\right)}{2\Delta u}\right) \cdot \frac{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot \left(u\left(D, \boldsymbol{z}'\right) + \Delta u\right)/2\Delta u\right) d\boldsymbol{z}'}{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u(D, \boldsymbol{z}')/2\Delta u\right) d\boldsymbol{z}'} \\
=&\exp\left(\frac{\varepsilon\Delta u}{2\Delta u}\right) \cdot \frac{\exp\left(\varepsilon\Delta u/2\Delta u\right) \int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u\left(D, \boldsymbol{z}'\right)/2\Delta u\right) d\boldsymbol{z}'}{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u(D, \boldsymbol{z}')/2\Delta u\right) d\boldsymbol{z}'} \\
=&\exp\left(\frac{\varepsilon}{2}\right) \cdot \exp\left(\frac{\varepsilon}{2}\right) = \exp(\varepsilon).
\end{aligned}$$

■

Another difference between the version in Dwork and Roth (2014) and ours, is that we explicitly show that the inequality follows from that $U\left(D', z'\right) \leq U\left(D, z'\right) + \Delta u$. In their version, it is directly stated that

$$\begin{aligned}
&\frac{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u\left(D', \boldsymbol{z}'\right)/2\Delta u\right) d\boldsymbol{z}'}{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u(D, \boldsymbol{z}')/2\Delta u\right) d\boldsymbol{z}'} \\
\leq&\frac{\exp\left(\varepsilon\Delta u/2\Delta u\right) \int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u\left(D, \boldsymbol{z}'\right)/2\Delta u\right) d\boldsymbol{z}'}{\int_{\boldsymbol{z}' \in \mathcal{Z}} \exp\left(\varepsilon \cdot u(D, \boldsymbol{z}')/2\Delta u\right) d\boldsymbol{z}'},
\end{aligned}$$

and we choose to include an extra step to add clarity.

Bayesian methods for ε-differentially private parameter estimation

Through the exponential mechanism we can introduce more complex differentially private mechanisms. We will now consider ways in which we can use

Bayesian posterior sampling to obtain differentially private point estimators of parameters. Intuitively, Bayesian methods coincide with the ideas behind differential privacy, as they introduce randomness through a (possibly noisy) prior distribution (Nikolenko, 2021; Wang, Fienberg and Smola, 2015).

Wang, Fienberg and Smola (2015) establish a connection between differential privacy and Bayesian sampling. They prove that if a single observation has a bounded contribution to the joint likelihood, then a mechanism $\mathcal{M}_B$ that draws a single sample from the posterior distribution is differentially private. We will now introduce their mechanism.

Say that $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is a set of observations, which we wish to fit to a model with parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. The prior distribution of the parameters is $p(\boldsymbol{\theta})$, and the likelihood of a single observation is $p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)$. From Bayes' theorem, we have that the posterior distribution of $\boldsymbol{\theta}$ is

$$p\left(\boldsymbol{\theta}|D\right) = \frac{p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)}{\int_{-\infty}^{\infty} p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right) d\boldsymbol{\theta}} \propto p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)$$

(Gelman, 2013).

Wang, Fienberg and Smola (2015) show that the exponential mechanism can be used to make the mechanism $\mathcal{M}_B$ is $\varepsilon-$differentially private. It requires a utility function $u(\boldsymbol{\theta}, D)$, so that the following holds:

$$p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right) = \exp\left(\frac{\varepsilon \cdot u(D, \boldsymbol{\theta})}{2\Delta u}\right).$$

Then it follows from Definition 2.5.4 that a sample from $p\left(\boldsymbol{\theta}|D\right)$ is $\varepsilon-$differentially private. The utility function is defined as

$$u(\boldsymbol{\theta}, D) = \log\left(p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)\right).$$

If we bound the contribution to the likelihood of a single observation $|\log\left(p\left(\boldsymbol{x_i}|\boldsymbol{\theta}\right)\right)| \leq B$ for all $i \in \{1, \ldots, n\}$ and $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, then $|u(D, \boldsymbol{\theta}) - u\left(D', \boldsymbol{\theta}\right)| \leq 2B$ for any two neighbouring data sets D and D'. It is straightforward to show that if we set $\varepsilon = 4B$, then

$$\exp\left(\frac{\varepsilon \cdot u(D, \boldsymbol{\theta})}{2\Delta u}\right) = \exp\left(\frac{4B \cdot log\left(p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right)\right)}{2 \cdot 2B}\right) = p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right).$$

The $4B-$differential privacy guarantee of $\mathcal{M}_B$ holds regardless of the choice of prior distribution $p(\boldsymbol{\theta})$ (Wang, Fienberg and Smola, 2015). We emphasise that the prior distribution must not depend on D. $\mathcal{M}_B$ does not utilise the randomness of the prior, as ε is only dependent on the bound of the likelihood.

Non-Bayesian work on differentially private point estimators include Smith (2008), where the estimate is the average of separate learners with added noise, and Dwork and Lei (2009) use the field of robust statistics to find differentially private parameter estimators. While more difficult to implement, they have the advantage of requiring less computational resources than Bayesian approaches.

Algorithm 4 $4B-$differentially private mechanism $\mathcal{M}_B$

Input:
$\mathcal{D} \leftarrow$ data set
$p(\boldsymbol{\theta}) \leftarrow$ prior distribution
$l\left(\mathcal{D}|\boldsymbol{\theta}\right)$, so that $\sup_{x \in D, \boldsymbol{\theta} \in \boldsymbol{\Theta}} |l\left(x|\boldsymbol{\theta}\right)| = B$
Output:
$\hat{\boldsymbol{\theta}}$

1: $p\left(\boldsymbol{\theta}|\mathcal{D}\right) \propto p(\boldsymbol{\theta})\, l\left(\mathcal{D}|\boldsymbol{\theta}\right)$
2: Draw a single sample $\hat{\boldsymbol{\theta}}$ from $p\left(\boldsymbol{\theta}|\mathcal{D}\right)$

(Wang, Fienberg and Smola, 2015, Algorithm 1).

Differentially private synthetic data.

We now have the necessary framework to explore differentially private synthesisers, but first we need to clarify the difference between a synthesiser model and a synthesiser mechanism. Following from Definition 2.1.5, a synthesiser $\mathcal{S}$ is a model and not a mechanism, since it takes a random noise signal as input and not a data set. However, $\mathcal{D}_s$ is fitted to a data set, and the fitting process is a mechanism which we call $\mathcal{M}_S$.

Algorithm 5 Synthesiser mechanism $\mathcal{M}_S : \mathcal{D}_r \to \mathcal{S}$

Input:
$\mathcal{D}_r \leftarrow$ real data set
Output:
$\mathcal{S} \leftarrow$ synthesiser fitted to $\mathcal{D}_r$

1: Fit $\mathcal{S}$ to $\mathcal{D}_r$

If $\mathcal{M}_S$ is $\varepsilon-$differentially private and outputs $\mathcal{S}$, then it is conventional to say that $\mathcal{S}$ and any synthetic data set $\mathcal{D}_s$ generated from $\mathcal{S}$ are ε-differentially private.

If we release a $\varepsilon-$differentially private synthesiser, then we know that it is impossible for adversaries to conduct attacks that reveal more information than the guarantee allows for, following from the the post-processing property. In practice, it is safer to only release a single data set, as it contains less information, but the property holds for the synthesiser as a whole. A non-differentially private synthesiser should in most cases not be released, because MIAs are simpler to perform when the attacker has access to the synthesiser directly and not just a single synthetic data set, as discussed in Section 2.5.2. That being said, a differentially private synthesiser is not immune against MIAs. However, the precision of an attack will decrease when ε decreases and the guarantee is stronger(C. Xu et al., 2019).

Nevertheless, it is a great advantage to be able to share a synthesiser more freely. If there is only one synthetic data set available, it might not contain the necessary information. For example, information system testing scenarios may require data which includes many outliers. By having direct access to a synthesiser, it is possible for analysts to generate as many outliers as needed. When a synthesiser permits conditioning, like CGAN, this is especially useful.

Another advantage to the post-processing property is that the $\varepsilon-$differential privacy guarantee also holds for any model fitted to a $\mathcal{D}_s$ generated from $\mathcal{S}$.

A requirement for differential privacy, as stated in Definition 2.5.1, is that $\mathcal{M}_S$ must be randomised. General non-differentially private synthesiser mechanisms are not required to be randomised, but a synthesiser model $\mathcal{S}$ is always probabilistic, as stated in Definition 2.1.5. As $\mathcal{M}_S$ is randomised and noise is added to the fitting process, we risk that the returned $\mathcal{S}$ has low utility, with the consequence that every $\mathcal{D}_s$ generated from $\mathcal{S}$ also has low utility. If we fit $\mathcal{S}$ multiple times to lower this risk, this activates the composition property. Each time we perform $\mathcal{M}_S$, we spend ε of the privacy budget. This is an example of how the privacy-utility trade-off applies to differentially private synthesisers. As a general rule, differentially private synthesisers have worse utility than non-differentially private synthesisers of the same architecture, and the utility decreases for smaller values of ε (Stadler, Oprisanu and Troncoso, 2022; Xie et al., 2018; C. Xu et al., 2019). However, Beaulieu-Jones et al. (2019) have seen a tendency that for very large data sets, differentially private synthesisers can perform better, because the differential privacy property reduces overfitting as a beneficial side effect.

There are many techniques for constructing a differentially private $\mathcal{M}_S$, and the details are outside the scope of this thesis. However, we will comment on some possibilities. If $\mathcal{S}$ is parametric, it can be fitted to the real data $\mathcal{D}_r$ by using the Bayesian approach in Algorithm 4. Examples of differentially private GANs are DP-GAN (Xie et al., 2018) where gradient cutting is used to add noise and PATE-GAN (Jordon, Yoon and M. v. d. Schaar, 2018) where noise is added by using the average signal from multiple models using the Private Aggregation of Teacher Ensembles (PATE) mechanism. We will return to differential privacy in Chapter 3 to study parametric mechanisms for MS-TTE data through the Bayesian approach.

2.6 Additional Privacy and Utility Evaluation Using Distance Metrics

In the previous sections, we have discussed privacy and utility evaluation separately. By utilising that there is a trade-off between privacy and utility, some methods can evaluate both. This is the case for distance metrics, which inherently assert both privacy and utility. This makes intuitive sense, as having $\mathcal{D}_r$ and $\mathcal{D}_s$ too close indicate that $\mathcal{D}_s$ is overfitted to $\mathcal{D}_r$, which causes privacy concern. Oppositely, if the data sets are too distant this can indicate poor utility.

2.6.1 Levels of Uncertainty in the Synthetic Data

We have so far discussed evaluation methods with respect to a single synthetic data set $\mathcal{D}_s$, compared to a single real data set $\mathcal{D}_r$, which is the standard approach. However, as the synthesiser is a probabilistic model, we will generate different synthetic data sets each time, which introduces a level of uncertainty. If we wish to evaluate the synthesiser, and not a specific synthetic data set, then we must generate multiple synthetic data sets and evaluate each of them separately.

In Section 2.5.3, we discussed how the fitting of the synthesiser can be a randomised mechanism, which we can think of as a second level of uncertainty. We return to this in Chapter 5.

One can also picture a third level of uncertainty, which stems from the uncertainty in the sample $\mathcal{D}_r^{(train)}$. If we are interested in synthetic data that describe the whole population, then a skewed sample of training data can introduce bias into the synthetic data. This may cause that $\mathcal{D}_s$ has low utility with respect to unseen real data. This can be controlled for by using a test set, $\mathcal{D}_r^{(test)}$. We will discuss these challenges in this section and how distance metrics can be applied to control them.

2.6.2 Distance to Closest Record (DCR)

Distances between data sets are not as simple to define as distances between individual points, wich are also called records. However, we can base the distance between the two data sets $\mathcal{D}_s$ and $\mathcal{D}_r$, $\|\mathcal{D}_s - \mathcal{D}_r\|$, on distances between individual points in the two data sets. For instance, we can for each point in $\mathcal{D}_s$ find the closest point in $\mathcal{D}_r$. To formalise we say that for each $\boldsymbol{x} \in \mathcal{D}_s$ we wish to find the distance to the closest $\boldsymbol{y} \in \mathcal{D}_r$:

$$dcr\left(\boldsymbol{x}, \mathcal{D}_r\right) = \min_{\boldsymbol{y} \in \mathcal{D}_r} \|\boldsymbol{x} - \boldsymbol{y}\|.$$

Zhao et al. (2021) refer to this measure as distance to closest record (DCR), which they use as a privacy metric only. The same idea is briefly discussed in Esteban, Hyland and Rätsch (2018). The choice of distance metric between the two points depends on the problem and data type, but $\|\boldsymbol{x} - \boldsymbol{y}\|$ needs to be the same metric for all pairs of points $\boldsymbol{x} \in \mathcal{D}_s$ and $\boldsymbol{y} \in \mathcal{D}_r$.

Example 2.6.1

Figure 2.1 displays 5 synthetic and 5 real randomly generated data points in a 2-dimensional space. For each synthetic data point we find the closest real data point using the Euclidean distance. They are connected with solid lines.

For some values of $\boldsymbol{x}_i \in \mathcal{D}_s$, we may have that $dcr\left(\boldsymbol{x}_i, \mathcal{D}_r\right) = 0$, which means that there exists exact copies of points in $\mathcal{D}_r$ in $\mathcal{D}_s$. One might be tempted to remove points from $\mathcal{D}_s$ that are too close to or equal to points in $\mathcal{D}_r$, because of a wish to protect the privacy of real data points. However, this can be too restrictive (El Emam, 2020). Firstly, as $\mathcal{S}$ is probabilistic, single points from $\mathcal{D}_s$ can appear to be very close, or possibly identical, to observations in $\mathcal{D}_r$ purely by chance. Randomly identical points are mainly a challenge in the discrete case. Secondly, when the number of observations in $\mathcal{D}_s$ increases, the points of $\mathcal{D}_s$ will cover more of the domain space, and the probability of generating synthetic data points that closely match points in its training data will increase. Thirdly, by doing so, one can construct "holes" in the domain space of $\mathcal{D}_s$, so that spaces with missing values in the reduced synthetic data set may indicate the position of observations in $\mathcal{D}_r$. A final point is that this could lead to utility loss, since the synthetic data no longer capture the whole data domain.

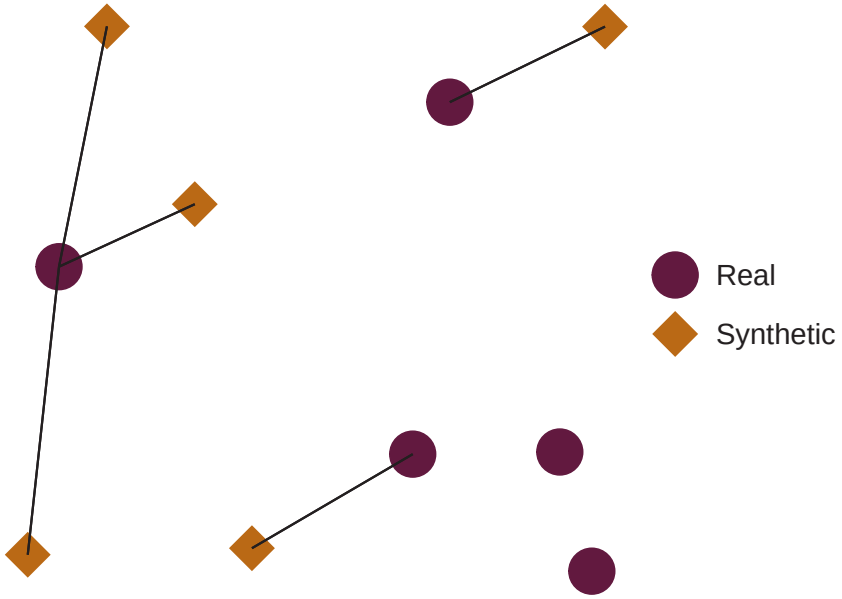


Figure 2.1: An illustration of DCR. We have five real and synthetic data points each. For each synthetic data point, we find the closest real data point.

Example 2.6.2 [Removing exact synthetic matches]

We will demonstrate how removing the exact matches can directly cause privacy violations. To do so, we use a two dimensional version of the Iris data set with variables petal length and sepal length, with 150 points as our $\mathcal{D}_r$. We then generate 15000 synthetic points using the R package *synthpop* (Nowok, Raab and Dibben, 2016). We use the default parametric generative method, *normrank*, which is a normal linear regression with the preservation of the marginal distributions. This is a randomised method. As we generate 100 times as many synthetic points as real points, we can expect that many of the synthetic points will overlap the real data by chance. The full synthetic Iris data set and the real Iris data set are displayed in Figure 2.2. As expected, it looks like we have overlap between the two, and we remove all exact matches and obtain a reduced synthetic Iris data set displayed in Figure 2.3. Since we have many points in $\mathcal{D}_s$, there are clearly visible gaps in the plot, where we can assume that there are real data points. We see that the real data points fill out these gaps in Figure 2.4. There are still some empty gaps where one might expect there to be real data points, illustrating that an adversary cannot know the exact position of real data points with certainty. Still, this discloses more information about $\mathcal{D}_r$ than intended.

While this is an exaggerated example, since we have many more synthetic than real data points, it illustrates that we should not blindly remove exact matches.

The DCR measure finds the distance between single points in a data set,

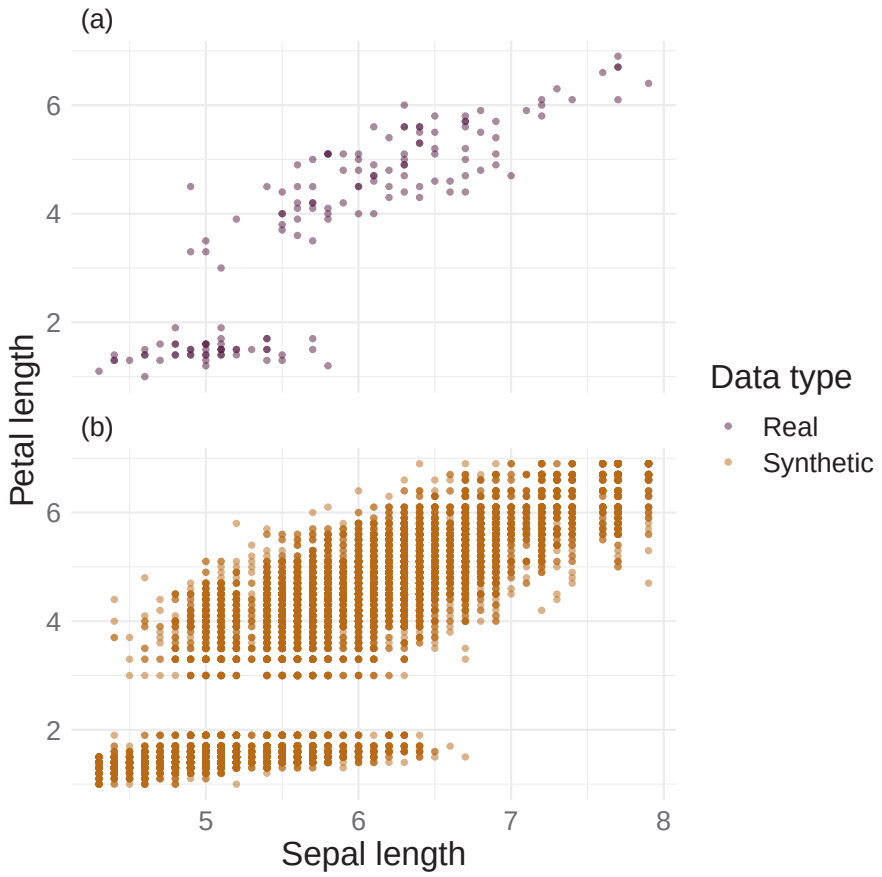


Figure 2.2: Scatter plot of the real (a) and full synthetic (b) Iris data sets used in Example 2.6.2.

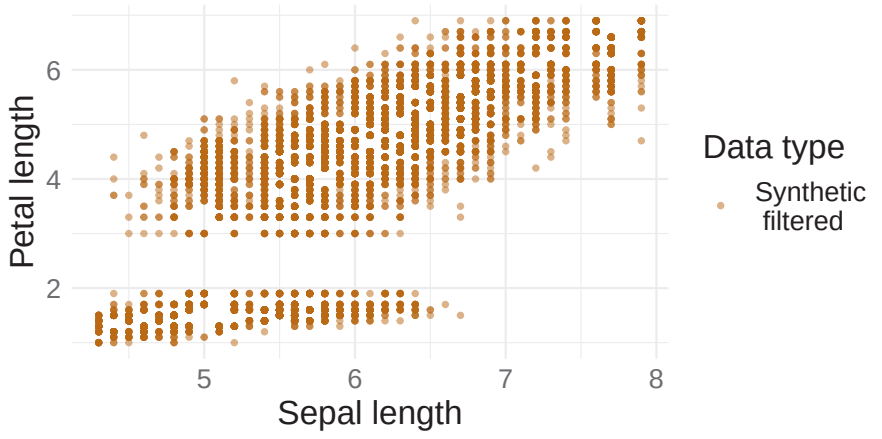


Figure 2.3: Scatter plot of the filtered synthetic Iris data, where all points that are exact matches of points in the real Iris data are removed.

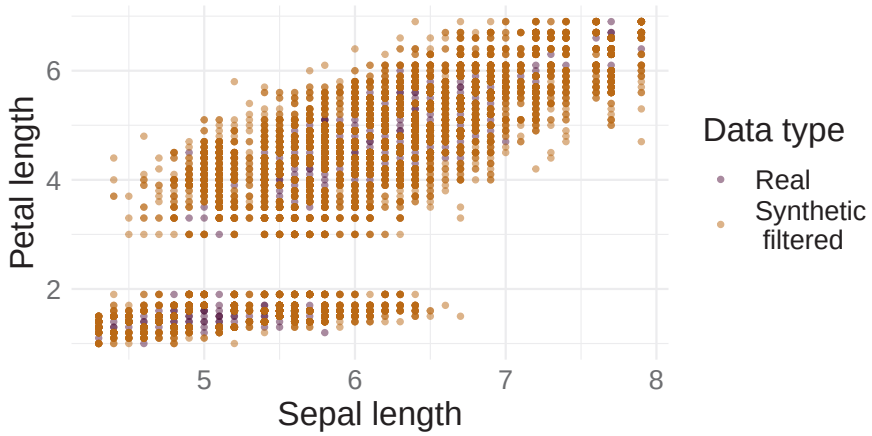


Figure 2.4: Scatter plot of the filtered synthetic Iris data and the real Iris data.

and we require a distance metric between two data sets. For each $\boldsymbol{x}_i \in \mathcal{D}_s$ for $i \in \{1, \ldots, n\}$ we can find their DCR values, and obtain the vector of size n

$$\boldsymbol{dcr}\left(\mathcal{D}_s, \mathcal{D}_r\right) = \left(dcr\left(\boldsymbol{x}_1, \mathcal{D}_r\right), \ldots, \ldots dcr\left(\boldsymbol{x}_n, \mathcal{D}_r\right)\right).$$

Zhao et al. (2021) use the 5th percentile of the vector $\boldsymbol{dcr}\left(\mathcal{D}_s, \mathcal{D}_r\right)$ as a metric, which we denote $\boldsymbol{dcr}_{0.05}$ and Guillaudeux et al. (2023) and Kotelnikov et al. (2022) use the median, $\boldsymbol{dcr}_{0.5}$. The latter gives an overall measure of the distance and is most useful for evaluating the general utility, while the former provides information about the distances to the points of a higher risk, which is useful from a privacy perspective. Still, if there are only a few $\boldsymbol{x}_j$ that are too close to $\mathcal{D}_r$, it might not be revealed by $\boldsymbol{dcr}_{0.05}$. Lower percentiles or the minimum value can be useful metrics, depending on the number of synthetic data points. For a general distance metric of the distance between two data sets with different quantiles α we use the notation

$$\boldsymbol{dcr}_\alpha\left(\mathcal{D}_s, \mathcal{D}_r\right), \alpha \in [0, 1].$$

Here, we will define percentiles of a vector in agreement with the *quantile* function in Core R (R Core Team, 2023), which is a weighted average of ordered elements.

Definition 2.6.3 [Percentiles]

For any vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, there exists an ordered vector $\left(x_{(1)}, \ldots, x_{(n)}\right)$, where $\left(x_{(i)} \leq x_{(i+1)}\right)$ for all $i \in \{1, \ldots, n-1\}$. The α percentile of $\boldsymbol{x}$, which we denote $\boldsymbol{x}_\alpha$ is defined as:

$$\boldsymbol{x}_\alpha = (\alpha + i - n\alpha)\, x_{(i)} + (1 + n\alpha - \alpha - i)\, x_{(i+1)},$$

where $i - 1 \leq n\alpha - \alpha < i$, and $\alpha \in [0, 1]$. Then, it follows that $\boldsymbol{x}_{\alpha=0} = \min(\boldsymbol{x})$ and $\boldsymbol{x}_{\alpha=1} = \max(\boldsymbol{x})$ (R Core Team, 2023).

Note that $\boldsymbol{dcr}_\alpha$ is non-symmetric, meaning that $\boldsymbol{dcr}_\alpha\left(D, D'\right)$ can be different from $\boldsymbol{dcr}_\alpha\left(D', D\right)$ when $D \neq D'$.

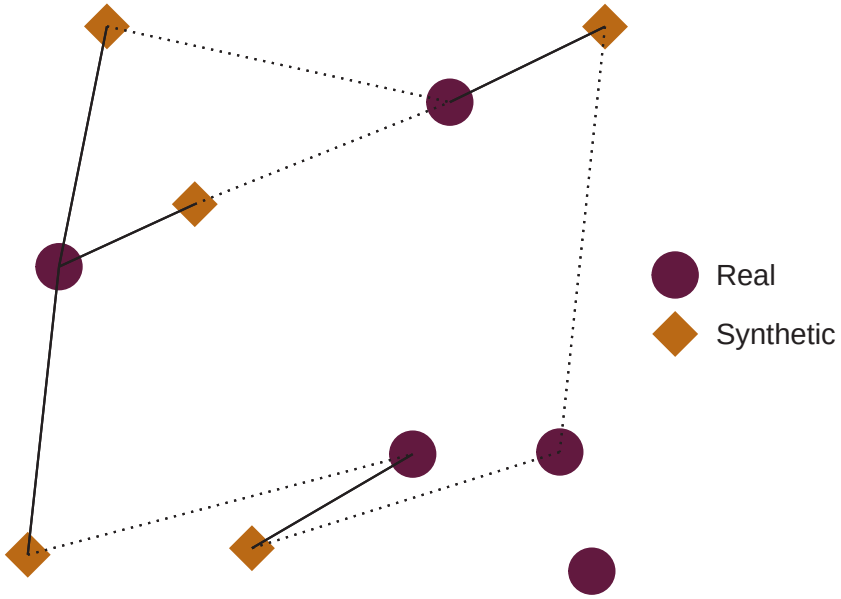


Figure 2.5: An illustration of NNDR using five real and synthetic data points. For each synthetic data point, we find the closest (solid line) and second closest (dotted line) real point.

2.6.3 Nearest Neighbour Distance Ratio (NNDR)

Zhao et al. (2021) introduces another distance metric called nearest neighbour distance ratio (NNDR). For each $\boldsymbol{x} \in \mathcal{D}_s$ we now consider the distances to the two closest matching points in $\mathcal{D}_r$, $\boldsymbol{y}^{1st}$ and $\boldsymbol{y}^{2nd}$. Then, we say that

$$nndr\left(\boldsymbol{x}, \mathcal{D}_r\right) = \begin{cases} \frac{\|\boldsymbol{y}^{1st} - \boldsymbol{x}\|}{\|\boldsymbol{y}^{2nd} - \boldsymbol{x}\|}, & \|\boldsymbol{y}^{1st} - \boldsymbol{x}\| > 0 \\ 0, & \|\boldsymbol{y}^{1st} - \boldsymbol{x}\| = 0 \end{cases}.$$

Example 2.6.4

In Figure 2.5 we have the same 5 synthetic and 5 real data points as in Example 2.6.1. In addition to connecting each synthetic data point to its closest real data point with a solid line, we also have a dotted line to the second closest real data point. The NNDR value for each synthetic point is the ratio between the solid and dotted line.

The metric $nndr\left(\boldsymbol{x}, \mathcal{D}_r\right)$ is bounded by $[0, 1]$, and low values show that $\boldsymbol{x}$ has more proximity to a specific point in $\mathcal{D}_r$ compared to the surrounding neighbourhood. This metric is useful to detect synthetic data points that are too close to real outliers. From a privacy perspective, this is more critical as opposed to when $\boldsymbol{x}$ has proximity to several data points in $\mathcal{D}_r$, which gives a NNDR score close to 1. We use the same notation as for DCR. For each $\boldsymbol{x}_i \in \mathcal{D}_s$ where $i \in \{1, \ldots, n\}$, we find their NNDR values, and get the vector

$$\boldsymbol{nndr}\left(\mathcal{D}_s, \mathcal{D}_r\right) = \left(nndr\left(\boldsymbol{x}_1, \mathcal{D}_r\right), \ldots, nndr\left(\boldsymbol{x}_n, \mathcal{D}_r\right)\right),$$

which like DCR is also non-symmetric. The α percentile of $\boldsymbol{nndr}$ is denoted

$$\boldsymbol{nndr}_\alpha\left(\mathcal{D}_s, \mathcal{D}_r\right), \alpha \in [0,1]\,.$$

As for DCR, Zhao et al. (2021) use the 5th percentile as evaluation metric. We argue that the choice of α should depend on the proportion of outliers in the data set. If the number of outliers is low, $\alpha = 0.05$ will not detect infringement of privacy for the outliers.

2.6.4 Distinguishing Between Random or Privacy Violating Proximity

The previous sections presented two different metrics of proximity between data sets, DCR and NNDR. As we have seen in Example 2.6.2, proximity can be caused by randomness in the synthesising process as well as because of privacy violation, and it is unwise to remove exact copies by default. However, the privacy of real points $\boldsymbol{y} \in \mathcal{D}_r$ with proximity to synthetic points $\boldsymbol{x} \in \mathcal{D}_s$ may still be at risk and should be subject to closer inspection.

Example 2.6.5 [Using distance metrics to expose privacy violation]

In this example we illustrate how distance metrics can be used to expose privacy violation using a staged and simple example. As previously discussed, if the Synthesising Distribution Assumption (Raab, Nowok and Dibben, 2016) holds and the true model $\mathcal{T}$ of the data is known, we can use it as a perfect synthesiser with respect to both privacy and utility. In such a case, say that $\mathcal{D}_r = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ is a sample from a known bivariate normal distribution, so that for each $\boldsymbol{y}_i$ where $i \in \{1, \ldots, n\}$ we have that

$$\boldsymbol{y}_i = \begin{pmatrix} y_{i,1} \\ y_{i,2} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}\right).$$

The first synthesiser, $\mathcal{S}_1$, samples from the same distribution, and we obtain a synthetic data set $\mathcal{D}_s^{(1)} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. We also fit $\mathcal{D}_r$ to another synthesiser, $\mathcal{S}_2$ which generates $\mathcal{D}_s^{(2)} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n\}$. For each $\boldsymbol{y}_i \in \mathcal{D}_r$, we have that

$$\boldsymbol{z}_i = \begin{pmatrix} y_{i,1} + \varepsilon_{i,1} \\ y_{i,2} + \varepsilon_{i,2} \end{pmatrix},$$

where $\varepsilon_{i,1}, \varepsilon_{i,2} \sim \mathcal{N}\left(0, 0.2^2\right)$. Instead of synthesising based on $\mathcal{D}_r$ as a whole, $\mathcal{S}_2$ adds noise signals to each individual real data point. Plots of the three data sets are displayed in Figure 2.6.

We evaluate the privacy and utility of $\mathcal{D}_s^{(1)}$ and $\mathcal{D}_s^{(2)}$ through the distance metrics presented in this section with $\alpha \in \{0.05, 0.5\}$. The results are displayed in Table 2.1. We see that the distance between the perfectly private synthetic data set $\mathcal{D}_s^{(1)}$ and $\mathcal{D}_r$ is greater than the distance between $\mathcal{D}_s^{(2)}$ and $\mathcal{D}_r$ for all four distance metrics.

In this simplified case it is clear that the structure of $\mathcal{S}_2$ inherently allows for privacy leakage from $\mathcal{D}_r$ into the synthetic data set $\mathcal{D}_s^{(2)}$. It only

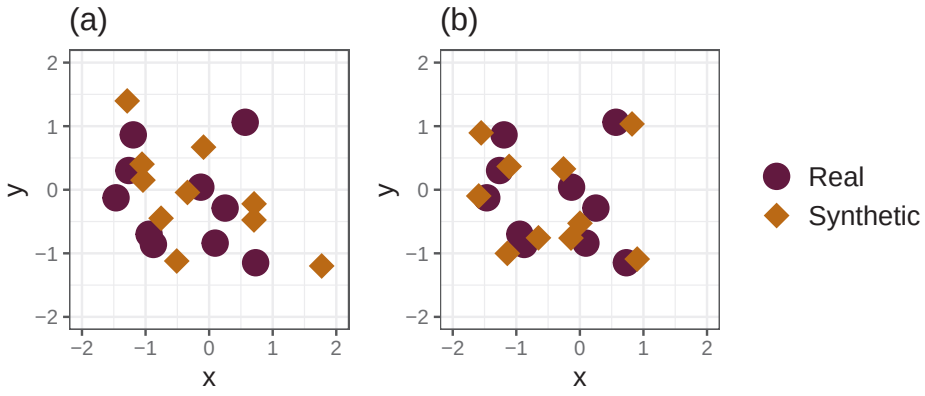


Figure 2.6: $\mathcal{D}_r$ and $\mathcal{D}_s^{(1)}$ (a) and $\mathcal{D}_r$ and $\mathcal{D}_s^{(2)}$ (b) from Example 2.6.5.

adds a small noise signal, which enables a direct link between real and synthetic observations. Still, we need to ask ourselves if the difference in distances in Table 2.1 in itself is enough to discard $\mathcal{S}_2$. The two distances are different, but we do not have the necessary information to evaluate if this difference is large enough to cause concern. This is especially true for the DCR metrics, because unlike NNDR they are not bounded and depend on the scale of the data. Another line of argument states that the proximity between $\mathcal{D}_r$ and $\mathcal{D}_s^{(2)}$ is not necessarily a fault of $\mathcal{S}_2$, but can rather be caused by the randomness of the synthesis process, which we discussed in the beginning of this section. A second synthetic data set from $\mathcal{S}_2$ might have sampled higher values of ε, which would result in larger distances.

Table 2.1: Distances in Example 2.6.5.

	$\boldsymbol{dcr}_{0.05}$	$\boldsymbol{dcr}_{0.5}$	$\boldsymbol{nndr}_{0.05}$	$\boldsymbol{nndr}_{0.0}$
$\|\mathcal{D}_s^{(1)} - \mathcal{D}_r\|$	0.228	0.455	0.409	0.568
$\|\mathcal{D}_s^{(2)} - \mathcal{D}_r\|$	0.145	0.251	0.197	0.399

To distinguish between proximity caused by randomness and privacy violating proximity, we propose four evaluation procedures using Monte Carlo methods. The procedures evaluate the synthesiser $\mathcal{S}$ rather than a single synthetic data set. We achieve this by repeatedly synthesising $k \in \{1, \ldots, K\}$ data sets $\mathcal{D}_s^k$ from a specified synthesiser $\mathcal{S}$ which is fitted to the training data $\mathcal{D}_r^{(train)}$. Because the synthesising process is randomised, the distance between

a synthetic data set $\mathcal{D}_s^k$ and a fixed real data set such as $\mathcal{D}_r^{(train)}$ is stochastic. We can find metrics such as the average distance between the synthetic data sets and the training data

$$\overline{\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|} = \frac{1}{K}\sum_{k=1}^{K}\|\mathcal{D}_s^{(k)} - \mathcal{D}_r^{(train)}\|.$$

This measure can be used to estimate how well the training data $\mathcal{D}_r^{(train)}$ is fitted to a synthesiser $\mathcal{S}$. As K increases, the variance caused by the randomisation of $\mathcal{S}$ will decrease. Moreover, the test set $\mathcal{D}_r^{(test)}$ can be used to evaluate if the synthetic data sets are closer to training data than unseen data. Our four evaluation procedures form a general framework that can be applied to other distance measures of a similar form to DCR and NNDR, where the distances between data sets are based upon distances between specific points. We will demonstrate these procedures using $\boldsymbol{dcr}_\alpha$ and $\boldsymbol{nndr}_\alpha$ for an $\alpha \in [0, 1]$.

Procedure 1

The first procedure follows the same principle as machine learning efficiency evaluation tests, as we discussed in Section 2.4.2, since a real hold-out test set is used for comparison. We wish to demonstrate how this principle can be applied to both privacy and utility evaluation through distance metrics. While similar ideas have been proposed by others (Guillaudeux et al., 2023), **we are not familiar with any prior work that makes use of repeated synthesis to distinguish between random and privacy violating proximity with DCR and NNDR as distance measures.**

In this procedure we test if the distance between a synthetic data set and its real training set is close to the distance between two real data sets. We formulate the following null hypothesis:

H_0 : The true mean of the distribution of the distance $\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$ is equal to $\|\mathcal{D}_r^{(test)} - \mathcal{D}_r^{(train)}\|$.

We test this hypothesis by generating $k \in \{1, \ldots, K\}$ synthetic data sets $\mathcal{D}_s^{(k)}$. To ensure a fair comparison, we require that $|\mathcal{D}_s^{(k)}| = |\mathcal{D}_r^{(test)}|$. This is necessary, because the distances between the data sets will decrease as the size of the data sets increase and the domain space is filled. If $\mathcal{D}_r^{(train)}$ is much closer to the synthetic data sets than to another real data set $\mathcal{D}_r^{(test)}$, then this indicates that $\mathcal{S}$ is overfitted to $\mathcal{D}_r^{(train)}$, which raises a privacy concern (Zhao et al., 2021). Contrary, if $\mathcal{D}_r^{(train)}$ is much closer to $\mathcal{D}_r^{(test)}$ than $\mathcal{D}_s$, then this indicates low utility. Therefore, the procedure can evaluate both the privacy and utility of $\mathcal{S}$.

The full procedure is described in Algorithm 6. As input we take a real training and test set, a synthesiser and how many synthetic data sets we wish to generate, which should depend on the size of $\mathcal{D}_r$ and the computational capacity. It outputs a p-value, mean distances between the synthetic and training data and the distance between the training and test data which is used as a benchmark. The latter is calculated in line 1. We then generate a synthetic data set $\mathcal{D}_s$ of the same size as $\mathcal{D}_r^{(train)}$ in line 3, and in line 4 we calculate the

distance between the newly generated $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$. This is then repeated K times, so that we have K different $\mathcal{D}_s$ and distances $\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$. In line 6 we perform a two-sided one-sample location test, which compares the location of $\boldsymbol{a}$ to b. The test can be parametric or non-parametric depending on the distribution of the distances. We will use the non-parametric Wilcoxon signed rank test (Devore, Berk and Carlton, 2021), as we do not want to assume that the distances follow a specified parametric distribution.

Algorithm 6 Procedure 1

Input:
$\mathcal{D}_r^{(train)} \leftarrow$ real training data set
$\mathcal{D}_r^{(test)} \leftarrow$ real test data set
$\mathcal{S} \leftarrow$ synthesiser
$K \leftarrow$ number of synthetic data sets
Output:
$p \leftarrow$ p-value
$\bar{\boldsymbol{a}} \leftarrow$ mean of distances between synthetic and training data
$b \leftarrow$ distance between test and training data

1: $b < -\|\mathcal{D}_r^{(test)} - \mathcal{D}_r^{(train)}\|$
2: **for** $k \in (1 : K)$ **do**
3: $\quad \mathcal{D}_s < -\mathcal{S}\left(\mathcal{D}_r^{(train)}, |\mathcal{D}_r^{(test)}|\right)$
4: $\quad a_i < -\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$
5: **end for**
6: $p <$ −p value of a two-sided one-sample location test $(\boldsymbol{a}, \boldsymbol{b})$

Procedure 2

The second procedure is also based upon machine learning efficiency evaluation as it uses a test set $\mathcal{D}_r^{(test)}$. Contrary to the first procedure, we will perform a one-sided hypothesis test, and we compare the distance between $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$ to the distance between $\mathcal{D}_s$ and $\mathcal{D}_r^{(test)}$. We formulate the following null hypothesis:

H_0 : The true mean of the distribution of the distance $\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$ is greater than or equal to the true mean of the distribution of the distance $\|\mathcal{D}_s - \mathcal{D}_r^{(test)}\|$.

This measures the privacy of $\mathcal{D}_s$ using a more direct approach than the first procedure, since we investigate if $\mathcal{S}$ is overfitted to $\mathcal{D}_r^{(train)}$. The procedure is displayed in Algorithm 7, which is similar to Algorithm 6. We now require that $|\mathcal{D}_r^{(train)}| = |\mathcal{D}_r^{(test)}|$, following the same reasoning as before. The synthetic data set can be of an arbitrary size, and this size is taken as an input. As both distances involve $\mathcal{D}_s$, they are both stochastic. Thus, we need to perform a two-sample one-sided test in line 6.

Algorithm 7 Procedure 2

Input:
$\mathcal{D}_r^{(train)} \leftarrow$ real training data set
$\mathcal{D}_r^{(test)} \leftarrow$ real test data set of the same size
$\mathcal{S} \leftarrow$ synthesiser
$K \leftarrow$ number of synthetic data sets
$n \leftarrow$ size of the synthetic data sets
Output:
$p \leftarrow$ p-value
$\bar{\boldsymbol{a}} \leftarrow$ mean distance between synthetic and training data
$\bar{\boldsymbol{b}} \leftarrow$ mean distance between synthetic and test data

1: **for** $k \in (1:K)$ **do**
2: $\quad \mathcal{D}_s < -\mathcal{S}\left(\mathcal{D}_r^{(train)}, n\right)$
3: $\quad a_i < -\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$
4: $\quad b_i < -\|\mathcal{D}_s - \mathcal{D}_r^{(test)}\|$
5: **end for**
6: $p <$ −p-value of a one-sided two-sample test $(\boldsymbol{a}, \boldsymbol{b})$

Procedure 3

This procedure based on the evaluation of Zhao et al. (2021), with the difference that they only considered a single synthetic data set. We extend their evaluation by applying repeated synthesis. This procedure differs from the first two, since it does not use separate test sets. This is beneficial when we need to use all of the real data for training. Instead of using a test set, the method of Zhao et al. relies on what we will call intra-data base distances, which we define in Definition 2.6.6.

Definition 2.6.6 [Intra-data set distance for DCR and NNDR metrics]

$IDD\left(\mathcal{D}\right)$ is the intra-data set distance of $\mathcal{D}$, as opposed to the inter-data set distance $\|\mathcal{D} - \mathcal{D}'\|$ between $\mathcal{D}$ and another data set $\mathcal{D}'$. It measures the homogeneity of a data set $\mathcal{D}$.

In the DCR case, the intra-data set distance is defined as:

$$\boldsymbol{dcr}\left(\mathcal{D}, \mathcal{D}\right) = \Big(dcr\left(x_1, \mathcal{D} \setminus \{x_1\}\right), \ldots, dcr\left(x_n, \mathcal{D} \setminus \{x_n\}\right)\Big)$$

for $\mathcal{D} = \{x_1, \ldots, x_n\}$, and likewise for NNDR.

The third procedure requires that $|\mathcal{D}_s^{(k)}| = |\mathcal{D}_r^{(train)}|$ for each $k \in \{1, \ldots, K\}$, and it tests the following null hypothesis:

H_0 : The true mean of the distribution of the distance $\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$ is equal to $IDD\left(\mathcal{D}_r^{(train)}\right)$.

This is similar to the first hypothesis in the sense that we wish to com-

pare the distance between $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$ to a distance between real data. Another similarity is that we compare a distance sample to a single test statistic and that this tests both the privacy and utility. We aim to generate synthetic data that is neither too close nor too distant from $\mathcal{D}_r^{(train)}$.

Procedure 4

Lastly, we will perform a procedure that evaluates the general utility of $\mathcal{D}_s$ by comparing the heterogeneity of $\mathcal{D}_s$ to that of $\mathcal{D}_r^{(train)}$. This procedure is also based on the evaluation of Zhao et al. (2021), with the difference that we use multiple synthetic data sets. The null hypothesis is

H_0 : The true mean of the distribution of the distance $IDD\left(\mathcal{D}_s\right)$ is equal to $IDD\left(\mathcal{D}_r^{(train)}\right)$.

The procedure requires that $|\mathcal{D}_s^{(k)}| = |\mathcal{D}_r^{(train)}|$ for each $k \in \{1, \ldots, K\}$. If $\mathcal{D}_s$ is more diverse than $\mathcal{D}_r^{(train)}$, meaning that $IDD\left(\mathcal{D}_s\right) > IDD\left(\mathcal{D}_r^{(train)}\right)$, then we have not correctly captured the distribution of $\mathcal{D}_r^{(train)}$. Oppositely, if $\mathcal{D}_r^{(train)}$ is more heterogeneous than $\mathcal{D}_s$ it also indicates low utility and can reveal mode collapse. A two-sided one-sample location test is performed to check if the difference in distance is significant. Algorithm 8 combines procedure 3 and 4. We can perform them together, because they both use a $50-50$ split and calculates the distance $IDD\left(\mathcal{D}_r^{(train)}\right)$. This is done in line 1 In line 4, we find the distance between $\mathcal{D}_s$ and $\mathcal{D}_r$ which is used in procedure 3, and in line 5 we find the intra-data set distance of $\mathcal{D}_s$, which we use in procedure 4. Lastly, in line 6 we perform the one-sample location test of procedure 3, and line 7 performs the one-sample location test of procedure 4.

We note that even when $|\mathcal{D}_s| = |\mathcal{D}_r|$, the comparison between $\boldsymbol{dcr}\left(\mathcal{D}_s, \mathcal{D}_r\right)$ and $\boldsymbol{dcr}\left(\mathcal{D}_s, \mathcal{D}_s\right)$ is unjust, because of a difference in intra- and inter-data set distances. The former compares each point in $\mathcal{D}_s$ to a data set of size $|\mathcal{D}_s| - 1$, while the latter compares each point to a data set which contains 1 more observation, which can lead to smaller distances. For larger data sets this difference becomes negligible, but if deemed necessary one option is to randomly remove a point from $\mathcal{D}_r^{(train)}$ before finding the distance in line 4.

In Chapter 4 we will explore the hypothesis framework for a specific data set. We are particularly interested in to see if it is beneficial to perform both procedure 1 and 3, or if they result in the same conclusions.

2.6.5 Notes on Single-Point Distance Metrics

While we have discussed distance metrics between data sets that rely on distance metrics between single points, we have not considered how single-point distances $\|\boldsymbol{x} - \boldsymbol{y}\|$ should be defined. The Euclidean distance is the most common option, but we are free to choose among other distance metrics. The choice should be made with the data type, number of dimensions and specific data set in mind. For example, rotated images can be far apart in Euclidean distance, despite looking very similar (Hastie, Tibshirani and Friedman, 2009).

Algorithm 8 Procedure 3 and 4

Input:

$\mathcal{D}_r^{(train)} \leftarrow$ training set
$\mathcal{S} \leftarrow$ synthesiser
$K \leftarrow$ number of synthetic data sets

Output:

$p_1, p_2 \leftarrow$ p-values
$\bar{\boldsymbol{a}} \leftarrow$ mean distance between synthetic and training data
$b \leftarrow$ intra-data set distance of the real data
$\bar{\boldsymbol{c}} \leftarrow$ mean intra-data set distance of the synthetic data

1: $b \leftarrow IDD\left(\mathcal{D}_r^{(train)}\right)$
2: **for** $k \in (1:K)$ **do**
3: $\quad \mathcal{D}_s < -\mathcal{S}\left(\mathcal{D}_r^{(train)}, |\mathcal{D}_r^{(train)}|\right)$
4: $\quad a_i < -\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$
5: $\quad c_i < -IDD\left(\mathcal{D}_s\right)$
6: **end for**
7: $p_1 <$ −p-value of a two-sided one-sample location test $(\boldsymbol{a}, b, \text{two-sided})$
8: $p_2 <$ −p-value of a two-sided one-sample location test $(\boldsymbol{c}, b)$

Finding k-nearest neighbours (kNN) can be challenging for high-dimensional data due to the curse of dimensionality (Hastie, Tibshirani and Friedman, 2009). Another challenge is finding the kNN of a point when the size of the data set is large, as the naive method measures the distance to every single point, which has complexity $O\left(n\right)$. More optimised methods such as kd-trees can speed up the search (Moore, 1990).

CHAPTER 3

Multi-State Time-to-Event Data

3.1 Survival Analysis Framework

Time-to-event data is, as the name spells out, data on the time (T) until an event occurs. The field of analysing such data is often called survival analysis, because of its typical applications to survival data. In this section, we will introduce the basic theoretical framework of survival analysis used within this thesis.

We begin by considering the PDF of T, which is denoted as $f(t)$. This density can be parametric, semi-parametric or non-parametric. Among parametric choices, the exponential or the Weibull distribution are commonly used. Semi-parametric alternatives include Cox-regression, while the Kaplan-Meier estimator is a non-parametric method (Aalen, Borgan and Gjessing, 2008). T can be discrete as well as continuous, but here we will only consider the continuous case.

A central function in survival analysis is $S(t)$, the survival function of T. Given a set of processes that all follow $f(t)$, the survival function gives us the expected proportion of processes that have not made a transition at time t, and is defined as:

$$S(t) = P(T > t) = \int_t^\infty f(t)\, dt \tag{3.1}$$

(Aalen, Borgan and Gjessing, 2008).

Next, we turn to the hazard function,

$$\alpha(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t} P\left(t \leq T \leq t + \Delta t | T \geq t\right)$$

(Aalen, Borgan and Gjessing, 2008), where $\alpha(t)$ gives us the conditional probability of transitioning at time t, given that a transition has not occurred before the time t.

The following property shows the relation between the 3 central functions $S(t)$, $\alpha(t)$ and $f(t)$:

$$f(t) = S(t)\,\alpha(t) \tag{3.2}$$

(Aalen, Borgan and Gjessing, 2008, (A.1)).

3.2 Weibull Regression

3.2.1 The Weibull Distribution

As stated, both the exponential and the Weibull distribution are used to model $f(t)$. Where the exponential distribution has a constant hazard function, the Weibull distribution allows for the hazard rate to change over time (Collett, 2015; Kalbfleisch, 2002). A changing hazard rate allows for more complex models, which motivates the use of the Weibull distribution in this thesis. The Weibull distribution is parameterised in different ways, and we will follow the notation used in Collett (2015). The distribution has shape (λ) and scale (σ) parameters and the PDF

$$f(t; \sigma, \lambda) = \lambda\sigma t^{\lambda-1} \exp\left(-\sigma t^\lambda\right), \; t, \lambda, \sigma > 0. \tag{3.3}$$

For completion, we derive the survival and hazard functions of the Weibull distribution. By the definition in Equation (3.1), we have that

$$S\left(t;\sigma,\lambda\right)=\int_{t}^{\infty}\lambda\sigma x^{\lambda-1}\exp\left(-\sigma x^{\lambda}\right)dx\,.$$

We solve the integral by substitution and use $u=\sigma x^{\lambda}$, so that $du\,/\,dx=\lambda\sigma x^{\lambda-1}$ and get

$$\int_{\sigma t^{\lambda}}^{\infty}\exp\left(-u\right)du=-exp\left(-u\right)\Big|_{\sigma t^{\lambda}}^{\infty}\,.$$

Then,

$$S\left(t;\sigma,\lambda\right)=\exp\left(-\sigma t^{\lambda}\right).$$

Following from Equation (3.2), the hazard function becomes

$$\alpha\left(t;\sigma,\lambda\right)=\lambda\sigma t^{\lambda-1}. \tag{3.4}$$

Note that the exponential distribution is a special case of the Weibull distribution when $\lambda=1$, and then the hazard function is constant for all t (Kalbfleisch, 2002).

Maximum likelihood parameter estimation

We can fit the shape and scale parameters of the Weibull distribution by maximum likelihood. For n independent and identically distributed processes $\boldsymbol{t}=(t_1,\ldots,t_n)$, the likelihood is

$$L\left(\sigma,\lambda;\boldsymbol{t}\right)=\prod_{i=1}^{n}f\left(t_i;\sigma,\lambda\right)=\prod_{i=1}^{n}\lambda\sigma t_i^{\lambda-1}\exp\left(-\sigma t_i^{\lambda}\right). \tag{3.5}$$

We take the log to obtain the log-likelihood

$$l\left(\sigma,\lambda;\boldsymbol{t}\right)=n\log\left(\lambda\right)+n\log\left(\sigma\right)+\left(\lambda-1\right)\sum_{i=1}^{n}\log\left(t_i\right)-\sum_{i=1}^{n}\left(\sigma t_i^{\lambda}\right). \tag{3.6}$$

The maximum value of the likelihood cannot be found analytically. The most common and straightforward way of finding the maximum likelihood estimates (MLEs) of a Weibull distribution is to apply numerical optimisation algorithms to the log-likelihood (3.6) (Aalen, Borgan and Gjessing, 2008).

3.2.2 Survival Regression Models

So far, we have considered a set of processes with equal distributions, meaning that each process has an equal probability of an event occurring at time t. In many situations, we may want t to depend on a set of covariates. We can do this by introducing survival regression models (Kalbfleisch, 2002). One class of survival regression models is the Cox proportional hazards model, which defines the hazard function in terms of a set of covariates $\boldsymbol{x}$ on the following form:

$$\alpha\left(t;\boldsymbol{x},\boldsymbol{\beta}\right)=\alpha_0\left(t\right)\exp\left(\boldsymbol{x}\boldsymbol{\beta}\right) \tag{3.7}$$

(Kleinbaum and Klein, 2005).

Cox proportional hazards models are semi-parametric, because $\alpha_0(t)$, which is called the baseline hazard, is non-parametric, and $\exp(\boldsymbol{x}\boldsymbol{\beta})$, which is called the exponential relative risk function, is parametric (Aalen, Borgan and Gjessing, 2008). The model holds the proportional hazards assumption, which states that only the baseline hazard, $\alpha_0(t)$, is dependent on t, whereas the covariates $\boldsymbol{x}$ are independent of t (Kleinbaum and Klein, 2005). The extended Cox model does not hold this assumption. Then, the hazard function is

$$\alpha\left(t; \boldsymbol{x}(t), \boldsymbol{\beta}\right) = \alpha_0(t) \exp\left(\boldsymbol{x}(t)\boldsymbol{\beta}\right) \tag{3.8}$$

(Kalbfleisch, 2002; Kleinbaum and Klein, 2005). Time-dependent covariates complicates the likelihood function (Kalbfleisch, 2002) and will not be considered here.

The proportional hazards name comes from that the coefficients can be interpreted in terms of the hazard rates. Say that two processes have the covariate vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ respectively. $x_{1,k} = x_{2,k}$ for all $k \in \{1, \ldots, K\} \setminus j$, and $x_{1,j} = x_{2,j} + 1$, where j is an arbitrary index in $\{1, \ldots, K\}$. That is, the two processes have the same values for all covariates except x_j, for which the first process has a value of one unit higher than the second process. Then the hazard rate of the two processes is

$$\frac{\alpha_0(t) \exp(\boldsymbol{x}_1\boldsymbol{\beta})}{\alpha_0(t) \exp(\boldsymbol{x}_2\boldsymbol{\beta})} = \frac{\exp\left(\beta_j\left(x_{2,j}+1\right)\right)}{\exp\left(\beta_j x_{2,j}\right)} = \exp\left(\beta_j\right)$$

(Aalen, Borgan and Gjessing, 2008).

This interpretation does not hold for the extended Cox model in Equation (3.8), since the covariates are not fixed for every t (Aalen, Borgan and Gjessing, 2008).

3.2.3 Weibull Proportional Hazards Regression (WPHR)

The Weibull proportional hazards regression (WPHR) model (Hosmer, 2008; Kundu, Darpe and Kulkarni, 2019; Z. Zhang, 2016) is a specific case of the Cox proportional hazards model (Kalbfleisch, 2002). We let $\log(\sigma)$ be the response of a linear regression model with K covariates $\boldsymbol{x}' = (1, x_1, \ldots, x_K)$ independent of t, and coefficients $\boldsymbol{\beta}' = (\beta_0, \beta_1, \ldots, \beta_K)$. Then, σ is defined as

$$\sigma = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_K x_K) = \exp\left(\boldsymbol{x}'\boldsymbol{\beta}'\right). \tag{3.9}$$

Using Equation (3.4), the hazard function of a WPHR model is

$$\alpha\left(t; \boldsymbol{x}', \lambda, \boldsymbol{\beta}'\right) = \lambda \exp\left(\boldsymbol{x}'\boldsymbol{\beta}'\right) t^{\lambda-1}.$$

If we want to write the hazard function in terms of the Cox proportional hazards model from Equation (3.7), we need to extract the intercept. Since it is not related to a covariate, β_0 is the same for each process, and must be a part of the baseline hazard. By redefining the vectors, so that $\boldsymbol{x} = (x_1, \ldots, x_K)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)$, we get that

$$\alpha(t; \boldsymbol{x}, \lambda, \beta_0, \boldsymbol{\beta}) = \lambda \exp(\beta_0) \exp(\boldsymbol{x}\boldsymbol{\beta})\, t^{\lambda-1} = \alpha_0(t, \lambda, \beta_0) \exp(\boldsymbol{x}\boldsymbol{\beta}),$$

where $\alpha_0(t, \lambda, \beta_0) = \lambda \exp(\beta_0) t^{\lambda-1}$. WPHR is fully parametric, since the baseline hazard is also parametric. The full PDF of a WPHR model is

$$f(t; \boldsymbol{x}, \lambda, \beta_0, \boldsymbol{\beta}) = \lambda \exp(\beta_0) \exp(\boldsymbol{x}\boldsymbol{\beta}) t^{\lambda-1} \exp\left(-\exp(\beta_0) \exp(\boldsymbol{x}\boldsymbol{\beta}) t^{\lambda}\right), \quad t, \lambda > 0.$$

3.2.4 Maximum Likelihood Parameter Estimation of a WPHR model

Because a WPHR model has coefficients $\boldsymbol{\beta}$ for each covariate in addition to a shape parameter λ and intercept β_0, the likelihood is more complex than the two-parameter Weibull distribution likelihood in Equation (3.5). Say we have $i \in \{1, \ldots, n\}$ processes with t_i and covariates $\boldsymbol{x}_i = (x_1, \ldots, x_K)$ independent of t_i. We also assume independence between each process. Further, we define $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. Then, the likelihood of a WPHR model is

$$L(\lambda, \beta_0, \boldsymbol{\beta}; \boldsymbol{t}, X) = \prod_{i=1}^{n} \lambda \exp(\beta_0) \exp(\boldsymbol{x}_i\boldsymbol{\beta}) t_i^{\lambda-1} \exp\left(-\exp(\beta_0) \exp(\boldsymbol{x}_i\boldsymbol{\beta}) t_i^{\lambda}\right). \tag{3.10}$$

Taking the log to obtain the log-likelihood we get

$$l(\lambda, \beta_0, \boldsymbol{\beta}; \boldsymbol{t}, X) = n \log(\lambda) + n\beta_0 + \sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{\beta} + (\lambda - 1) \sum_{i=1}^{n} \log(t_i) - \exp(\beta_0) \sum_{i=1}^{n} \exp(\boldsymbol{x}_i\boldsymbol{\beta}) t_i^{\lambda}. \tag{3.11}$$

Software such as the R package *flexsurv* (Jackson, 2016) can find the maximum of Equation (3.11) numerically to obtain the MLEs of the parameters.

3.3 Censored and Uncensored Data

Example 3.3.1 [NAV right-censoring]

At NAV, each registered job seeker gets a digital activity plan to assist them with getting a job (NAV, 2023). As an hypothetical example, say that NAV is conducting a trial of a new activity plan system. The study consists of two groups of job seekers, where one group gets access to a new system, and the other is a control group. They are interested in the time it takes to get a job, and the aim of the study is to see if the job seeking process is quicker for the group using the new system. Over an observation period, they will record the time t until a participant gets a job. When the observation period is over, there may still be some job seekers left, which makes the data incomplete. An incomplete observation may signify that the job seeker finds a job at some later point, but it can also mean that the job seeker will never find a job. The only information disclosed by the data is that they could not find a job during the observation period. Even so, this is valuable information. Instead of discarding these observations, they

can be kept as what is known as right-censored observations (Kalbfleisch, 2002).

Time-to-event data often contains censored observations, which can either be because the event did not happen before the observation period is over, as in Example 3.3.1 (right-censoring), or because the event happened before the observation period began (left-censoring). Complete observations are referred to as uncensored observations (Aalen, Borgan and Gjessing, 2008).

The censored observations can still be used to find the MLE of the parameters $\boldsymbol{\theta}$ of a survival regression model with probability distribution $f(t; \boldsymbol{x}, \boldsymbol{\theta})$, and we will demonstrate this for the right-censored case. We define a vector of independent random variables $\boldsymbol{T} = (T_1, \ldots, T_n)$ with covariates $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ that follow the distribution $f(t; \boldsymbol{x}, \boldsymbol{\theta})$. For each T_i for $i \in \{1, \ldots, n\}$, we have a censoring time c_i which is independent of T_i. In the case where $T_i > c_i$, the transition time is not observed, and we say that the observation is right-censored. Using the definition of a survival function $S(t; \boldsymbol{\theta})$ in Equation (3.1), the contribution to the likelihood is then $P\left(T_i > c_i | \boldsymbol{x}_i, \boldsymbol{\theta}\right) = S(c_i; \boldsymbol{x}_i, \boldsymbol{\theta})$. When $T_i \leq c_i$, we can observe T_i directly, and the contribution to the likelihood is $f(T_i; \boldsymbol{x}_i, \boldsymbol{\theta})$ (Aalen, Borgan and Gjessing, 2008).

To simplify the definition of the likelihood, we include an indicator vector $\boldsymbol{d} = \left(\mathbb{I}(T_1 \leq c_1), \ldots, \mathbb{I}(T_n \leq c_n)\right)$, meaning that if $d_i = 0$, the event is right-censored, and if $d_i = 1$, the event is observed. In addition, we will use the notation $t_i = \min(T_i, c_i)$ (Aalen, Borgan and Gjessing, 2008). We can define the likelihood as

$$\begin{aligned} L\left(\boldsymbol{\theta} | \boldsymbol{t}, \boldsymbol{d}, X\right) &= \prod_{i:d_i=1} f(t_i; \boldsymbol{x}_i, \boldsymbol{\theta}) \prod_{i:d_i=0} S(t_i; \boldsymbol{x}_i, \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} f(t_i; \boldsymbol{x}_i, \boldsymbol{\theta})^{d_i} S\left(t_i | \boldsymbol{x}_i, \boldsymbol{\theta}\right)^{1-d_i} \\ &= \prod_{i=1}^{n} \alpha(t_i; \boldsymbol{x}_i, \boldsymbol{\theta})^{d_i} S(t_i; \boldsymbol{x}_i, \boldsymbol{\theta}) \end{aligned} \tag{3.12}$$

(Jackson, 2016; Aalen, Borgan and Gjessing, 2008). The last equality follows from Equation (3.2). The log-likelihood is

$$l\left(\boldsymbol{\theta} | \boldsymbol{t}, \boldsymbol{d}, X\right) = \sum_{i_1}^{n} \left[d_i \log\left(\alpha(t_i; \boldsymbol{x}_i, \boldsymbol{\theta})\right) + \log\left(S(t_i; \boldsymbol{x}_i, \boldsymbol{\theta})\right) \right]. \tag{3.13}$$

Here we assume that the censoring times $\boldsymbol{c}$ are known, but it is also possible to let the censoring times be independent samples from a distribution $g(c_i; \boldsymbol{\psi})$ which is independent of $f(t_i; \boldsymbol{x}_i, \boldsymbol{\theta})$, and using the likelihood of the joint distribution (Aalen, Borgan and Gjessing, 2008). Because this is currently not supported by the R package *flexsurv* (Jackson, 2016) we will not consider it here.

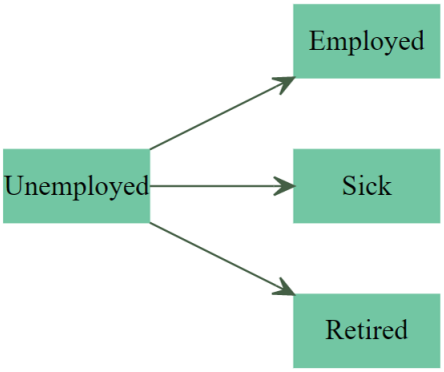


Figure 3.1: Illustration of a competing risks model with four states and three competing legal transitions.

3.4 Multi-State Models

3.4.1 Competing-Risks Models

So far, we have considered time-to-event data with only one possible outcome. In Example 3.3.1, the participants go from unemployed to employed, and the transition can only happen once. If NAV wants to model other possible outcomes, such as going from unemployed to receiving sickness benefits or retirement pension, they can use a competing risks model (Kalbfleisch, 2002). Then, an event is considered as a transition between states. In Example 3.3.1, we have two states, *unemployed* and *employed*, and t is the time until an individual transitions from the first to the second state. In a competing risks model we can add states like *sick* and *retired*, as shown in Figure 3.1. Multi-state models are commonly illustrated in this manner, with boxes represents states, and arrows representing the permitted transitions (Andersen and Keiding, 2002). In this thesis, we will use a simple approach to competing-risks models, which assumes independence between the transitions. In many cases, this is an oversimplification of reality. Still, this is a common assumption, as it lowers the computational cost of finding the combined likelihood (Jackson, 2022).

We generalise by saying that from the starting state r, it is possible to make $LT \in \mathbb{N}$ legal transitions into the states $q_1, \ldots, q_{LT}$. Each transition $r \to q_j$, for each $j \in \{1, \ldots, LT\}$, is modelled as an independent survival regression model $f_j\left(t; \boldsymbol{x}, \boldsymbol{\theta}_j\right)$ (Jackson, 2022), analogous to the models discussed in Section 3.3. The name competing-risks motivates an interpretation of LT events with independent risks, which compete to become the first to occur. When one event has taken place, it is not possible for the same process to experience any of the remaining events. The independence assumption requires that there are are no constraints to the parameters of the survival regression models across transitions (Jackson, 2022). That is, $\boldsymbol{\theta}_j \perp \boldsymbol{\theta}_{j'}$ for every $j \neq j'$.

The combined likelihood of a competing-risks model with censoring is the product of the likelihoods of each independent transition, which is on a similar form as Equation (3.12). This follows from the independence assumption. Say that we have n observations $(t_i, s_i, d_i, \boldsymbol{x}_i)$ for each $i \in \{1, \ldots, n\}$. Each

$s_i \in \{1, \ldots, LT\}$ indicates a legal transition $r \to q_{s_i}$, $t_i \in [0, \infty)$ indicates the transition or censoring time, $d_i \in \{0, 1\}$ is a censoring indicator and $\boldsymbol{x}_i$ is a covariate vector. Since each transition has its own likelihood, one might be tricked to believe that the observation $(s_i, t_i, d_i, \boldsymbol{x}_i)$ with $s_i = j$ should only contribute to the likelihood of the transition $r \to q_j$. However, we will also use this observation to calculate the likelihoods of the other transitions, because we can think of the observation as a right-censoring of the remaining transitions at time t_i (Jackson, 2022). In this way, we utilise the information that the other events did not happen before time t_i.

The likelihood of a competing risks model is

$$L\left(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT} | \boldsymbol{t}, \boldsymbol{s}, \boldsymbol{d}, X\right) = \prod_{j=1}^{LT} \left[\prod_{i:s_i=j} \left[f_j\left(t_i; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right)^{d_i} S_j\left(t_i; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right)^{1-d_i} \right] \prod_{i:s_i \neq j} S_j\left(t_i; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right) \right]. \tag{3.14}$$

When we use this notation, any observation $(s_i, t_i, d_i, \boldsymbol{x}_i)$ where $d_i = 0$ has an arbitrary value of s_i, because the event is censored and none of the transitions occurred. As a way to explicitly state that each observation contributes to the likelihoods of all the transitions, and to remove ambiguity for censored observations, we can use a *long format* representation (Wreede, Fiocco and Putter, 2011). In this new representation, we write that each $i \in \{1, \ldots, n\}$ consists of three vectors. First, a covariate vector covariate vector $\boldsymbol{x}_i$, which is defined as before. Next, we have a vector $\boldsymbol{d}_i$ of length LT, which keeps track of the censored transitions. If $d_{i,j} = 1$, then this means that the ith observation experienced the jth transition. We either have that $\sum_{j=1}^{LT} d_{i,j} = 1$, which means that all transitions except one is censored, or $\sum_{j=1}^{LT} d_{i,j} = 0$, if all the transitions are censored. Finally, we have the vector $\boldsymbol{t}_i$, which consists of LT repetitions of the transition time t_i from before. It indicates the censoring or transition times of each transition. While this notation may seem overly uncompressed, it will become useful for the models in Section 3.4.2. Further, it simplifies the likelihood of a competing-risks model:

$$L\left(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT} | T, D, X\right) = \prod_{j=1}^{LT} \prod_{i=1}^{n} \left[f_j\left(t_{i,j}; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right)^{d_{i,j}} S_j\left(t_{i,j}; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right)^{1-d_{i,j}} \right], \tag{3.15}$$

where $T = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n)$ and $D = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n)$.

The R package *flexsurv* (Jackson, 2016) can be used to find the MLE estimates of the parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT}$. In this section we have defined the likelihood based on Equation (1) in Jackson (2016), which states that the likelihood of the full model is the product of the joint likelihoods of each transition. However, we have expanded the definition of the likelihoods in Equation (3.14) and Equation (3.15) by using the notation for censored likelihoods used in Section 3.3, so that we explicitly show how the censoring is used in competing-risks models.

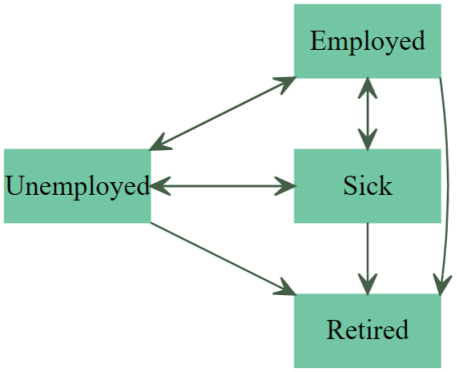


Figure 3.2: Illustration of a multi-state model with three transient states and one absorbent state. There are nine legal transitions in total.

3.4.2 Multi-State Time-to-Event (MS-TTE) Models

Competing-risks models are not complex enough for many types of data, as they do not allow for a single process to experience multiple events. For example, the model in Figure 3.1 is not suitable for a process where an individual goes from unemployed to employed, and then after some time retires. This becomes achievable with the introduction of multi-state models (Kalbfleisch, 2002), which allows a single process to make multiple transitions between states. Such a model is illustrated in Figure 3.2.

A multi-state time-to-event (MS-TTE) model is used to model the transition times of stochastic processes that transition between states according to a multi-state model (Meira-Machado et al., 2009). Like before, we will here consider the continuous-time case. Similar to how we represented competing-risks models as a combination of several time-to-event models, we will use the common approach of modelling MS-TTE data as a combination of competing-risks models (Jackson, 2022).

In a multi-state model, we say that a state is transient if there are one or more legal transitions from that state, and otherwise we say that the state is absorbent (Andersen and Keiding, 2002), meaning that a process will terminate once it enters an absorbent state. We will use the cause-specific hazards of competing risks framework (Jackson, 2022) to create MS-TTE models, in which an MS-TTE model consists of a sub-model for each transient state, and where each sub-model is a competing-risks model. To do so, we first need to expand our notation. Say that we have R transient states with indices $r \in \{1, \dots, R\}$, that each have $lt_r \in \mathbb{N}$ legal transitions. Then, the total number of legal transitions j in the multi-state model is $LT = \sum_{r=1}^{R} lt_r$.

Example 3.4.1

The model in Figure 3.2 is a simplified version of the MS-TTE model used by Gran et al. (2015), where a multi-state model was applied to NAV data on sickness absence. Our model has 4 states, where *unemployed*, *employed*

and *sick* are transient and *retired* is absorbent. There are 3 legal transitions from all transient states, which makes the total number of legal transitions $LT = 9$.

If we assume independence between each competing-risks model, and as before, assume independence between each transition, then the likelihood of an MS-TTE model follows the same principle as Equation (3.14), where the combined likelihood is the product of independent joint likelihoods of each legal transition (Jackson, 2022). Each legal transition has its own transition index $j \in \{1, \ldots, LT\}$ and an independent time-to-event regression model $f_j\left(t; \boldsymbol{x}, \boldsymbol{\theta}_j\right)$.

The difference between the likelihood of a competing-risks model and an MS-TTE model is that each observation can consist of multiple uncensored transitions. Another difference is that each uncensored transition is not paired with censored transitions of all the remaining transitions. This is just the case for the transitions that are in the same competing risks sub-model, meaning transitions that go *from* the same state. In Figure 3.2, the transition *Employed* $\rightarrow$ *Unemployed* should only contribute to the likelihood of this transition, and to the likelihoods of *Employed* $\rightarrow$ *Sick* and *Employed* $\rightarrow$ *Retired*, which are the competing transitions.

The most straightforward way to represent this is by expanding the long format representation used in Equation (3.15). Since we can have more than one uncensored transition, we also introduce a transition vector $\boldsymbol{s}_i$ for each observation i, which holds the indices of all the censored and uncensored transitions. It follows the structure of the multi-state model, so that each element in $\boldsymbol{s}_i$ that represents an uncensored transition is followed by elements representing the censored transitions of the competing-risks sub-model. The length will vary for each observation, and we say that it has length m_i. The vector $\boldsymbol{t}_i$ is also of length m_i and stores the transition or censoring times. This is managed by the censoring indices in vector $\boldsymbol{d}_i$. T, D and X are defined as before, and $S = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n)$. The combined likelihood of an MS-TTE model becomes

$$\begin{aligned} &L\left(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT} | T, S, D, X\right) \\ &= \prod_{j=1}^{LT} \prod_{i=1}^{n} \prod_{l:s_{i,l}=j} \left[f_j\left(t_{i,l}; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right)^{d_{i,l}} S_j\left(t_{i,l}; \boldsymbol{x}_i, \boldsymbol{\theta}_j\right)^{1-d_{i,l}} \right]. \end{aligned} \tag{3.16}$$

Besides likelihood calculations, it is most practical to use a short format representation of each observation i, where $\boldsymbol{s}_i$ only contains the uncensored transitions. That is, $\boldsymbol{s}_i$ is the observed state trajectory. On short-format, the scalar d_i indicates if the observation i is censored. If $d_i = 0$, then we have not observed a final transition into an absorbent state. This means that the last element in $\boldsymbol{t}_i$ is a censoring time, while all other elements are transition times. If we also know the structure of the multi-state model, then this can easily be extended to the long format representation. The distinction between short and long format is much used for MS-TTE data (Meira-Machado et al., 2009; Wreede, Fiocco and Putter, 2011). We will state which format we use in each case.

3.5 Time Homogeneity and Inhomogeneity

We have defined the random variable T as the time-to-event. This is unambiguous for a model with only one possible transition or a competing-risks model, since each process can only experience one event (Jackson, 2016). For multi-state models we need to clarify what we consider as the starting time t_0. One option is to define it as the time the process enters its current state, meaning that we reset the time after each transition. This is assumed to be the case in Equation (3.16). Such a model is called a clock-reset or a time homogeneous model. Another alternative is to define t_0 as the beginning of the process, and this is called a clock-forward or time inhomogeneous model (Incerti and Jansen, 2021; Jackson, 2016; Kalbfleisch, 2002).

3.5.1 Clock-Reset Models

Clock-reset models are simpler to both fit and use, because we do not need to use conditional probabilities and we can use the same probability distributions $f_j\left(t; \boldsymbol{x}, \boldsymbol{\theta}_j\right)$ for each transition. When we fit transition data to a clock-reset model, we will for each observation only consider the time spent in the current state before the next transition occurs, and all information about the total length of the process is disregarded.

Say that we have an observation i with covariate vector $\boldsymbol{x}_i$ that has made m_i transitions $\boldsymbol{s}_i = \left(s_{i,1}, \ldots, s_{i,m_i}\right)$ at times $\boldsymbol{t}^* = \left(t^*_{i,1}, \ldots, t^*_{i,m_i}\right)$. None of the transitions are censored, meaning that the vectors are on short-format. We consider the times t^* to be the time since the beginning of the process, which means that $0 \leq t^*_{i,1} \leq \ldots \leq t^*_{i,m_i}$. In a clock-reset model, we redefine the times to $t_{i,l} = t^*_{i,l} - t^*_{i,l-1}$ for all $l \in \{1, \ldots, m_i\}$, and we can use the likelihood function in Equation (3.16) as before.

Under the assumption that $\boldsymbol{\theta}_j \perp \boldsymbol{\theta}_{j'}$ for every $j \neq j'$, we have that each $t_{i,l} \perp t_{i,l'}$ for every $l \neq l'$. This is why clock-reset models often are called semi-Markov models (Jackson, 2022), because the model only considers the time spent in the current state, and the transition history and the time since the beginning of the process is disregarded. Thus, the model is memoryless and follows the Markov assumption.

A clock-reset model is not a good fit in cases where the transition probabilities are dependent on the total time of the process and the transition history. This requires that the transition probabilities are adjusted each time a process reenters a state, which is not possible for a clock-reset model.

Example 3.5.1

For the multi-state model in Figure 3.2 we may wish to increase the risk of transitioning to state *retired* as the total time passes, or to include that the risk of transitioning to state *sick* depends upon prior sick leave history. Then a clock-reset model with the Markov property is not suitable.

One solution that sets aside the Markov assumption is to include covariates which describe the history of the process (Andersen and Keiding, 2002). Explicit examples include the total time of the process, number of times the states have

been entered and proportion of time spent in each state, while more implicit covariates can for example be the age of an individual. While it is possible to have time-dependent covariates, which we saw in Equation (3.8), this is outside the scope of this thesis. A common workaround is to let the covariates refer to the time of entry. Then a covariate like age can refer to age at the last transition, and the covariate is assumed constant as long as a process remains in the same state (Andersen and Keiding, 2002; Jackson, 2016). We call this semi-time-dependent covariates.

3.5.2 Clock-Forward Models

In a clock-forward model, the process does not have a memory of the previous states as before, but the transition probabilities depends on both the total time of the process and the time since the last transition. Clock-forward models rely on left-truncation, which is commonly also referred to as delayed entry (Aalen, Borgan and Gjessing, 2008).

Say that a process enters a state r at time t_l. From r, it is possible to transition into lt_r other states, and each transition has an independent probability distribution $f_j(t;\boldsymbol{\theta})$. In a clock-reset model, t is reset to 0, but a clock-forward model considers the probability of transitioning at time $t \geq t_l$ given that the process entered the state at time t_l. We do this by conditioning on $t \geq t_l$: $f_j\left(t|t \geq t_l;\boldsymbol{\theta}_j\right)$ (Jackson, 2016).

In Figure 3.3 (a) we see a Weibull PDF with parameters $\lambda = 1$ and $\sigma = 1.5$. In (b), the PDF is left-truncated at $t = 1$ and in (c) the PDF is left-truncated at $t = 2$. Consider these distributions as the probability of a process making a transition from a state r at time t. In (a), (b) and (c), the process entered state r at the delayed times $t = 0$, $t = 1$ and $t = 2$ respectively. We observe that with later entering times, the probability of staying in state r for a short time increases.

A clock-forward model requires an adjusted likelihood, so that the distributions are left-truncated at the last transition time. Like in Equation (3.16), we will use a long format representation to define the likelihood. Say that each observation $i \in \{1,\ldots,n\}$ consists of m_i censored and uncensored transitions and has a covariate vector $\boldsymbol{x}_i$, a vector of transitions $\boldsymbol{s}_i = \left(s_{i,1},\ldots,s_{i,m_i}\right)$, $\boldsymbol{t}_i = \left(t_{i,1},\ldots,t_{i,m_i}\right)$ contains the transition times, counting from the time of the beginning of the process, and the vector $\boldsymbol{d}_i = \left(d_{i,1},\ldots,d_{i,m_i}\right)$ indicates if a transition is censored or not. As before, $T = (\boldsymbol{t}_1,\ldots,\boldsymbol{t}_n)$, $D = (\boldsymbol{d}_1,\ldots,\boldsymbol{d}_n)$, $S = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n)$ and $X = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$. The likelihood of a clock-forward model is

$$\begin{aligned} &L\left(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_{LT}|T,S,D,X\right) \\ &= \prod_{j=1}^{LT}\prod_{i=1}^{n}\prod_{l:s_{i,l}=j}\left[f_j\left(t_{i,l}|t_{i,l}\geq t_{i,l-1};\boldsymbol{x}_i,\boldsymbol{\theta}_j\right)^{d_{i,l}} S_j\left(t_{i,l}|t_{i,l}\geq t_{i,l-1};\boldsymbol{x}_i,\boldsymbol{\theta}_j\right)^{1-d_{i,l}}\right]. \end{aligned} \tag{3.17}$$

3.5.3 Maximum Likelihood Estimation using *flexsurv*

The R package *flexsurv* (Jackson, 2016) offers support for multi-state models, both for clock-reset and clock-forward models. The package makes it

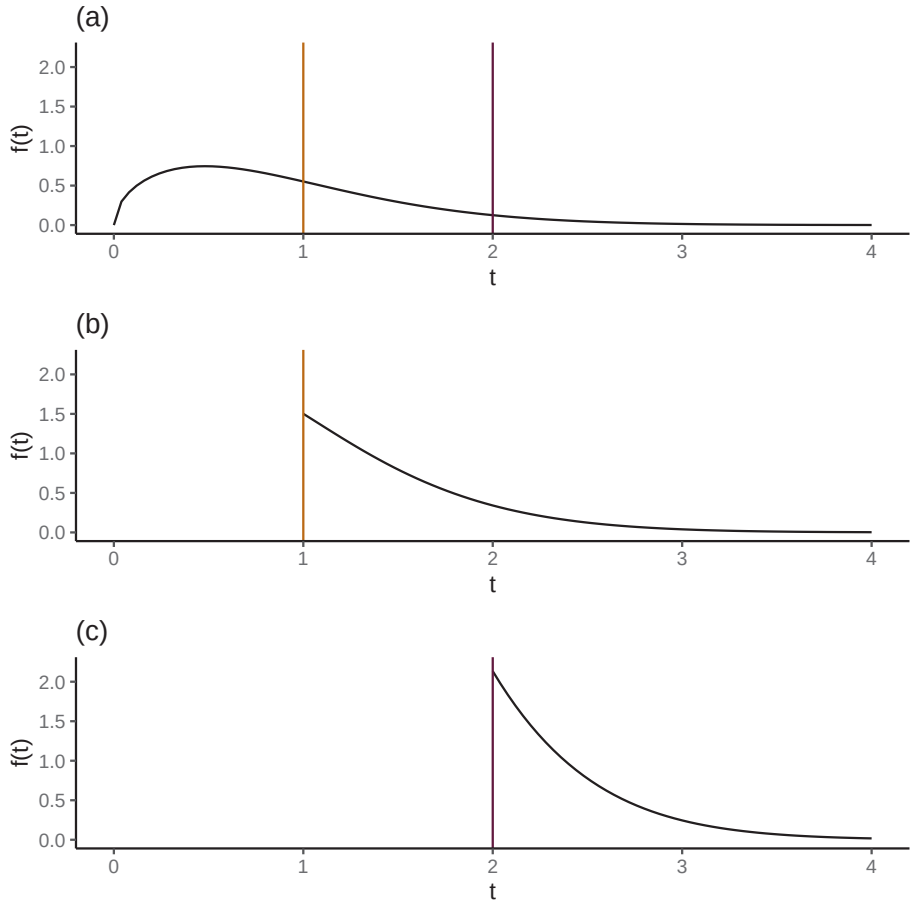


Figure 3.3: A Weibull PDF $f(t)$ with $\lambda = 1.5$ and $\sigma = 1$ (a), $f(t)$ left-truncated at $t = 1$ (b) and $f(t)$ left-truncated at $t = 2$ (c).

straightforward to find the MLEs of Equation (3.16) and Equation (3.17) numerically, and it supports many parametric and non-parametric distributions for each valid transition $j \in \{1, \ldots, LT\}$ (Jackson, 2022). We refer to the documentation for details on the optimisation methods used. When clock-reset and clock-forward models are fitted using *flexsurv* in later chapters, we will use a WPHR model for every transition. When each transition in an MS-TTE model follows a WPHR model, then we will call it a WPHR MS-TTE model.

3.6 Synthesising from a Multi-State Time-to-Event Model

Once we have fitted an MS-TTE model to data, we can use the fitted model to generate new synthetic processes. Given a covariate vector $\boldsymbol{x}$ and a fitted model with the parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT})$, we can draw new synthetic processes with state trajectories $\boldsymbol{s}$ and corresponding transition times $\boldsymbol{t}$, both on the short format.

3.6.1 Clock-Reset Synthesis

Algorithm 9 is a generalisation of Algorithm 1 in Incerti and Jansen (2021), which is customised to a specific domain. The algorithm demonstrates how we can synthesise from a clock-forward MS-TTE model with at least one absorbent state. In addition to the model parameters and a covariate vector, it also requires a starting state, which must be non-absorbent. In lines $1-2$ we set an index and the starting time of the total time $\boldsymbol{t}$ to 0. The synthesis procedure continues as long as the current state is transient, and from each transient state we sample from a competing-risks model. We do so by sampling a transition time from each valid transition in line 5, and the transition that takes place first is selected in line 7. In Section 3.5.1 we discussed the use of semi-time-dependent covariates. If any such covariates are included in the model, they are updated in line 8. Then, the new time and states are added to the vectors $\boldsymbol{t}$ and $\boldsymbol{s}$ in lines $9-10$. This is repeated until an absorbent state is reached.

Algorithm 9 Synthesise a Process from a Clock-Reset Multi-State Time-to-Event Model

Input:

$\boldsymbol{x} \leftarrow$ covariates of the synthetic process
$s_0 \leftarrow$ starting state of the synthetic process
$\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT}) \leftarrow$ parameters of an MS-TTE clock-reset model with LT transitions

Output:

$\boldsymbol{t} \leftarrow$ transition times of the synthetic process (short format)
$\boldsymbol{s} \leftarrow$ state trajectory of the synthetic process (short format)

1: $l \leftarrow 0$
2: $t_l \leftarrow 0$
3: **while** s_l is transient **do**
4: **for** each valid transition j from state s_l **do**
5: Sample t'_j from $f_j\left(t; \boldsymbol{\theta}_j, \boldsymbol{x}\right)$
6: **end for**
7: $j' \leftarrow \text{argmin}_j\ t'_j$
8: Update any semi-time-dependent covariates with $t'_{j'}$
9: $t_{l+1} \leftarrow t_l + t'_{j'}$
10: $s_{l+1} \leftarrow$ new state following transition j'
11: $l \leftarrow l + 1$
12: **end while**

3.6.2 Clock-Forward Synthesis

For clock-forward models we use Algorithm 10, which is an updated version of Algorithm 9. In this version, t is drawn from a left-truncated distribution in line 5 (Incerti and Jansen, 2021). The parameters $\boldsymbol{\theta}$ of a clock-forward model are also different, because of the difference in likelihood functions, as we saw in Equation (3.16) and Equation (3.17). We will not use semi-time-dependent covariates in clock-forward models, because the passing of time is already considered by not

resetting the time upon entering a new state. However, we can have covariates like the age at t_0, which are time-invariant.

Algorithm 10 Synthesise a Process from a Clock-Forward Multi-State Time-to-Event Model

Input:

$\boldsymbol{x} \leftarrow$ covariates of the synthetic process
$s_0 \leftarrow$ starting state of the synthetic process
$\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{LT}) \leftarrow$ parameters of an MS-TTE clock-forward model with LT legal transitions

Output:

$\boldsymbol{t} \leftarrow$ transition times of the synthetic process (short format)
$\boldsymbol{s} \leftarrow$ state trajectory of the synthetic process (short format)

1: $l \leftarrow 0$
2: $t_l \leftarrow 0$
3: **while** s_l is transient **do**
4: **for** each valid transition j from state s_l **do**
5: Sample t'_j from $f_j\left(t | t > t_l; \boldsymbol{\theta}_j, \boldsymbol{x}\right)$
6: **end for**
7: $j' \leftarrow \text{argmin}_j\ t'_j$
8: $t_{l+1} \leftarrow t'_{j'}$
9: $s_{l+1} \leftarrow$ new state following transition j'
10: $l \leftarrow l + 1$
11: **end while**

3.6.3 Complete Synthesis

In the documentation of the R package *hesim* (Incerti and Jansen, 2021; Incerti and Jansen, 2022), their Algorithm 1 is used to simulate a set of new samples. Their MS-TTE model is fitted to real data, but we still categorise their approach as simulation and not synthesis, a distinction we discussed in Section 2.1.2. Both their approach and our Algorithm 9 and Algorithm 10 are conditioned on on covariates $\boldsymbol{x}$ and a starting state s_0. Incerti and Jansen (2022) simulate the variables $(\boldsymbol{x}, s_0)$ from a distribution which has *not* been estimated from real data, which is why we categorise it as a simulation. That being said, the distinction between synthetic and simulated data is not entirely coherent, because we can also view Algorithm 9 and Algorithm 10 as conditional synthesisers, like the CGAN and CTGAN discussed in Section 2.2.1.

In Chapter 4 we will demonstrate how we can obtain fully synthetic MS-TTE data. There we will use a real data set $\mathcal{D}_r$ to fit both an MS-TTE model, and a synthesiser for the tabular data $(\boldsymbol{x}, s_0)$ Combined, they become an MS-TTE synthesiser. This is to the best of our knowledge **the first formulation of a synthesiser for MS-TTE data**. We sketch out the fitting process in Algorithm 11. The real data $\mathcal{D}_r$ has the same structure and long-format representation as the data we used to fit the likelihoods in Equation (3.16) and Equation (3.17). The returned synthetic data $\mathcal{D}_s$ is also on long format, but it does not contain any censored points, meaning that all the synthetic trajectories end in an absorbent state. In line 1, we fit an arbitrary tabular synthesiser $\mathcal{S}_{tab}$

to the covariates $X^{(r)}$ and starting states $S_0^{(r)}$ of the real data $\mathcal{D}_r$. Next, we fit the MS-TTE synthesiser $\mathcal{S}_{ms}$ in line 2. Here we have the choice between a clock-reset $\mathcal{S}_{ms}^{(cr)}$ or a clock-forward synthesiser $\mathcal{S}_{ms}^{(cf)}$. The synthesiser $\mathcal{S}_{ms}^{(cr)}$ has parameters $\boldsymbol{\theta}$ that are fitted using the likelihood in Equation (3.16) and follows Algorithm 9. The synthesiser $\mathcal{S}_{ms}^{(cf)}$ has parameters $\boldsymbol{\theta}$ that are fitted using the likelihood in Equation (3.17) and follows Algorithm 10. In line 4 we sample a synthetic point from $\mathcal{S}_{tab}$, and we condition on this point as we sample from $\mathcal{S}_{ms}$ in line 5. We repeat for the desired number of synthetic points. Finally, we transform $\mathcal{D}_s$ into long format in line 7. This is done by expanding the vectors $\boldsymbol{t}_i$ and $\boldsymbol{s}_i$ of each synthetic point according to the multi-state model of $\mathcal{D}_r$. In addition, we form a censoring indicator vector $\boldsymbol{d}_i$. Full details on the implementation can be found in Appendix B.

We say that $\mathcal{S}^{(cr)} = \left(\mathcal{S}_{tab}, \mathcal{S}_{ms}^{(cr)}\right)$, and $\mathcal{S}^{(cf)} = \left(\mathcal{S}_{tab}, \mathcal{S}_{ms}^{(cf)}\right)$, and that $\mathcal{S}^{(cr)}$ and $\mathcal{S}^{(cf)}$ are full MS-TTE synthesisers.

Algorithm 11 Complete Synthesis of Multi-State Time-to-Event Data

Input:
$\mathcal{D}_r \leftarrow$ real training data consisting of $X^{(r)}, S^{(r)}, T^{(r)}, D^{(r)}$ on long format
$n \leftarrow$ number of synthetic samples

Output:
$\mathcal{D}_s \leftarrow$ synthetic data consisting of $X^{(syn)}, S^{(syn)}, T^{(syn)}, D^{syn}$ on long format

1: Fit a tabular synthesiser $\mathcal{S}_{tab}$ to $X^{(r)}$ and $S_0^{(r)}$
2: Fit an MS-TTE synthesiser $\mathcal{S}_{ms}$ with parameters $\boldsymbol{\theta}$ to $X^{(r)}, S^{(r)}, T^{(r)}$ and $D^{(r)}$
3: **for** $i \in \{1, \ldots, n\}$ **do**
4: Sample $\boldsymbol{x}_i^{(syn)}$ and $s_{i,0}^{(syn)}$ from $\mathcal{S}_{tab}$
5: Sample $\boldsymbol{t}_i^{(syn)}, \boldsymbol{s}_i^{(syn)}$ from $\mathcal{S}_{ms}$
6: **end for**
7: $X^{(syn)}, S^{(syn)}, T^{(syn)}, D^{syn} \leftarrow$ long format$\left(\mathcal{D}_r, X^{(syn)}, S^{(syn)}, T^{(syn)}\right)$

Algorithm 11 can only synthesise uncensored data points, because even when $\mathcal{D}_s$ is on long format and contains censored transitions, we still only have synthetic points with full trajectories that end in an absorbent state. This is useful in many cases, as complete and uncensored data contain more information than data that includes censored processes. However, if we wish to generate a synthetic data set $\mathcal{D}_s$ which is equal to $\mathcal{D}_r$ also in terms of proportion of censored data, this is a limitation. As a possible extension to censored data, one can also condition on a censoring time in Algorithm 9 and Algorithm 10, which is also drawn from a distribution fitted to $\mathcal{D}_r$. Norcliffe et al. (2023) do this in their synthesiser SurvivalGAN for survival data with one event. Recall from Section 2.2.3 that SurvivalGAN has a similar structure to the synthesisers $\mathcal{S}^{(cr)}$ and $\mathcal{S}^{(cf)}$ proposed above, as it combines a tabular synthesiser with a survival regression function. We note that we developed our synthesisers before Norcliffe et al. (2023) published their synthesiser, and consequently our synthesisers are independent of their work.

3.7 Differential Privacy for Survival Analysis

In Section 2.5.3 we discussed how differential privacy requires randomised mechanisms with no exact mapping between input data and output. Maximum likelihood estimation does not fall into this category, which means that a synthesiser $\mathcal{S}_{ms}$ used in Algorithm 11 with parameters $\boldsymbol{\theta}_{ML}$ cannot be differentially private. In this section, we will apply the Bayesian mechanism $\mathcal{M}_B$ introduced in Section 2.5.3 to a Weibull Regression MS-TTE model, which enables differentially private synthesis of MS-TTE data. To achieve this, we also consider differential privacy for Weibull Regression models.

3.7.1 Prior Work on Differential Privacy for Survival Data

Because survival data often include health data and other sensitive domains, there has been several approaches to differentially private survival models. This includes non-parametric approaches such as the Kaplan-Meyer estimator (Bonomi, Jiang and Ohno-Machado, 2020) and semi-parametric models (Nguyên and Hui, 2018). The latter mechanism is similar in structure to the mechanism $\mathcal{M}_B$ introduced by Wang, Fienberg and Smola (2015), in the sense that it draws a single sample of parameters from a posterior distribution with bounded likelihood. Nguyên and Hui (2018) apply this idea to semi-parametric proportional hazards models, such as Cox regression, and proportional odds models. They are semi-parametric with non-parametric baseline functions that are modelled with I-splines.

We continue our focus on the parametric approach, and specifically Weibull regression. While we are not familiar with any prior work on differentially private Weibull regression, Nguyên and Hui (2017) were first to propose a mechanism that provides differentially private parameter estimations of the shape (λ) and scale (σ) parameters of a Weibull distribution (which we defined in Equation (3.3)).

3.7.2 Differential Privacy for Weibull Regression

Our proposal is to to apply the mechanism of Nguyên and Hui (2018) to a WPHR model, which is also a proportional hazards model. We can discard the use of I-splines since the WPHR model is fully parametric. In this section, we present our novel differentially private WPHR model in this section, which to our knowledge is **the first development of differentially private Weibull regression**.

First, we will discuss the mechanism presented in Nguyên and Hui (2018) in greater detail. Recall the exponential mechanism and utility functions which we covered in Definition 2.5.4. Further recall from Section 2.5.3 that the mechanism $\mathcal{M}_B$ used a bound on the utility function so that $\sup_{x\in D,\boldsymbol{\theta}\in\boldsymbol{\Theta}}|l\left(x|\boldsymbol{\theta}\right)| = B$. Nguyên and Hui (2018) have a similar procedure, and they use a sanitiser function to bound the likelihood:

$$C\left(x;\eta\right) = \frac{\varepsilon}{2\eta}\begin{cases} x & -\eta \leq x \leq 0 \\ 0 & x > 0 \\ -\eta & x < -\eta \end{cases}. \tag{3.18}$$

The name sanitiser function comes from data sanitisation, which is the process of preparing a data set for release with respect to privacy protection (Sramka et al., 2010).

Using a notation consistent with the likelihood in Equation (3.12), we have a data set $\mathcal{D} = \{(\boldsymbol{x}_1, t_1, d_1), \ldots, (\boldsymbol{x}_n, t_n, d_n)\}$. For each $i \in \{1, \ldots, n\}$ there is a covariate vector $\boldsymbol{x}_i$, time t_i, and indicator d_i, so that if $d_i = 1$, then t_i is a transition, and otherwise t_i is a right-censoring time. In Nguyên and Hui (2018), the utility function is

$$u(\boldsymbol{\theta}, \mathcal{D}) = \log\big(p(\boldsymbol{\theta})\big) + \sum_{i=1}^{n} C\left(l(\boldsymbol{\theta}; \boldsymbol{x}_i, t_i, d_i) - d_i \frac{\eta}{2}; \eta\right), \tag{3.19}$$

where $p(\boldsymbol{\theta})$ can be any prior, and the log-likelihood is as defined in Equation (3.13). The sanitiser function bounds the contribution of a single observation to the joint likelihood, and it has a hyper-parameter $\eta > 0$, which we will return to shortly. Given that $p(\boldsymbol{\theta})$ does not depend on $\mathcal{D}$, we have from Definition 2.5.2 and the bounds of C that $\Delta u = \frac{\varepsilon}{2\eta}\big(0 - (-\eta)\big) = \frac{\varepsilon}{2}$.

We note that if the sanitiser function was replaced by the likelihood directly, then Equation (3.19) is the log of the unnormalised posterior distribution $p(\boldsymbol{\theta}) \prod_{i=1}^{n} p\big(\boldsymbol{\theta}|\boldsymbol{x}_i, t_i, d_i\big)$. We are interested in the posterior distribution with a modified likelihood, which is

$$p\big(\boldsymbol{\theta}|\mathcal{D}\big) \propto p(\boldsymbol{\theta}) \exp\left(\sum_{i=1}^{n} C\left(l(\boldsymbol{\theta}; \boldsymbol{x}_i, t_i, d_i) - d_i \frac{\eta}{2}; \eta\right)\right) \tag{3.20}$$

(Nguyên and Hui, 2018).

Similar to to Algorithm 4, a single sample from the distribution $p\big(\boldsymbol{\theta}|\mathcal{D}\big)$is $\varepsilon-$differentially private. This follows from Definition 2.5.4. We will use Markov chain Monte Carlo (MCMC) methods to sample from the normalised distribution (Gelman, 2013).

Notes on the hyper-parameters ε and η

The two hyper-parameters η and ε control the privacy in different ways through the sanitiser function in Equation (3.18). The sanitiser function sets positive log-likelihood values to 0, so that the contribution of observations with high likelihood values are cut. Take note of that this only applies to uncensored observations. This is because censored observations only contribute to the joint likelihood through the survival function, and $S(t) \in [0, 1]$, meaning that $\log\big(S(t)\big) \leq 0$. The hazard function $\alpha(t)$ does not have an upper bound, meaning that the log-likelihood can become positive for uncensored observations. This is why an offset $-d_i\eta/2$ is used in Equation (3.19) (Nguyên and Hui, 2018) and Equation (3.20), and the value of η determines how much uncensored observations can contribute to the joint likelihood.

Nguyên and Hui (2018) recommends to define $\eta = \log(n)$. They do not properly clarify why this is, but as we have previously discussed the level of noise needed to uphold a certain privacy level decreases as n increases. This makes it clear that η must increase when n increases, because then there is less of a concern that a single observation will influence the joint likelihood considerably. In Equation (3.18) we see that η also controls the lower bound. If

the likelihood of an observation is low, it will be set to η. It is important that η is not too low, because then few observations are permitted to contribute with the full magnitude of their likelihoods values to the joint likelihood, and the signal will be reduced. As η increase, the offset also increases. Thus, too high values of η will lead to that the contributions to the likelihood of the uncensored observations are greatly reduced.

The overall level of privacy is controlled by ε, following Definition 2.5.1, and ε does not depend on η. The privacy is ensured by limiting how much the likelihood can influence the posterior distribution. If $\varepsilon = 0$, then only the prior distribution will be used, and as ε increases, the likelihood will dominate, assuming that n is large enough. If ε is large, but η is too small, then there is less noise from the prior, but the signal from the joint likelihood will only depend upon a few observations. Oppositely, if ϵ is small and η is large, then most of the observations will contribute to the joint likelihood, but the prior will dominate the posterior distribution. We discuss the tuning of these parameters for a specific data set in Chapter 5.

Differentially private Weibull proportional hazards regression

We now replace the likelihood $l(\boldsymbol{\theta}; \boldsymbol{x}_i, t_i, d_i)$ with the likelihood of a WPHR model from Equation (3.10) with adjustments to tailor for censored data, according to Equation (3.13). Then, a single sample from

$$p\left(\lambda, \beta_0, \boldsymbol{\beta} | \mathcal{D}\right) \propto$$
$$p(\lambda, \beta_0, \boldsymbol{\beta}) \exp\left(\sum_{i=1}^{n} C\left(d_i\left(\log(\lambda) + \beta_0 + \boldsymbol{x}_i\boldsymbol{\beta} + (\lambda - 1)\log(t_i)\right)\right.\right.$$
$$\left.\left. - \exp(\beta_0)\exp\left(\boldsymbol{x}_i\boldsymbol{\beta}'\right) t_i^{\lambda} - d_i\frac{\eta}{2}; \eta\right)\right)$$

is $\varepsilon-$differentially private estimates of the parameters $\lambda, \beta_0, \boldsymbol{\beta}$ of a WPHR model.

3.7.3 Differential Privacy for MS-TTE Models.

In the previous section we considered differential privacy for survival models with only one transition per observation and an MS-TTE model can have multiple transitions per observation. We argue that it is necessary to have a lower bound per transition for an observation with many transitions, to ensure that each observation is subject to the same bound. **We present a differentially private MS-TTE synthesiser mechanism** which meets this requirement. Here, we will only consider clock-reset models, because they are easier to implement. However, this can also be extended to clock-forward models by adjusting the likelihoods accordingly.

As before, say we have $j \in \{1, \ldots, LT\}$ legal transitions in total, and we aim to fit a time-to-event model for each transition separately, similar to Equation (3.16). Each transition has a privacy budget ε/LT, which will make the combined MS-TTE model $\varepsilon-$differentially private. This follows from the composition property of differential privacy, which we considered in Section 2.5.3. The data set $\mathcal{D}$ is on long format, and we split it into into LT disjoint

subsets $\mathcal{D}_j$, each only containing data on transition j. In $\mathcal{D}_j$ we have a total of $i \in \{1, \ldots, n_j\}$ observations, each of which have made $m_{i,j}$ censored and uncensored transitions j. Each subset is defined as

$$\mathcal{D}_j = \left\{ \left(\boldsymbol{t}_{1,j}, \boldsymbol{x}_1, \boldsymbol{d}_{1,j} \right), \ldots, \left(\boldsymbol{t}_{n_j,j}, \boldsymbol{x}_{n_j}, \boldsymbol{d}_{n_j,j} \right) \right\}.$$

The adjusted utility function is defined with respect to each subset $\mathcal{D}_j$:

$$u\left(\boldsymbol{\theta}, \mathcal{D}_j\right) = \log\left(p\left(\boldsymbol{\theta}\right) + \sum_{i=1}^{n_j} C\left(\frac{1}{m_{i,j}} \sum_{l=1}^{m_{i,j}} l\left(\boldsymbol{\theta}; \boldsymbol{x}_i, \boldsymbol{t}_{i,j}, \boldsymbol{d}_{i,j}\right); \eta \right) \right). \quad (3.21)$$

Consequently, the bound is set with respect to the total contribution to the joint likelihood of each observation, and not each transition separately. This is done by entering an averaged likelihood of each observation into the sanitiser function. We find that when each state in the multi-state model has more than one possible legal transition, like in Figure 3.2, there is no added benefit in including an offset. As we take the average of several transitions, both uncensored and censored, high uncensored contributions to the likelihood are balanced out by the uncensored contributions. This may be necessary to adjust if the hazard function can return very high values for some t.

This is a novel approach which can be used for any parametric MS-TTE model, and in Chapter 5 we will demonstrate it on a specific data set using a WPHR MS-TTE model. There we continue the discussion on hyper-parameter tuning.

CHAPTER 4

Performing MS-TTE Data Synthesis

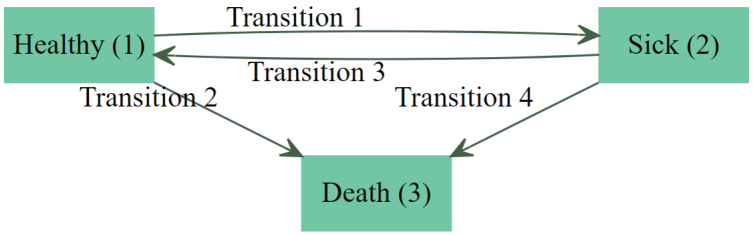


Figure 4.1: Illustration of an illness-death model with recovery.

4.1 Liver Cirrhosis Data

In this chapter we will apply the methods of Chapter 2 and 3 to real MS-TTE data. Because of the sensitivity of NAV data, we will use an open-access data set. The aim of this chapter is to perform a demonstration of the synthesis and evaluation methods, with transparent results that enables reproducibility. However, as we only use a single data set, this is not intended as a comparison study of the different methods.

This real data set is collected from the health domain, and will henceforth be referred to as liver cirrhosis data. The data is used as example data in the R packages *hesim* (Incerti and Jansen, 2021) and *mstate* (Wreede, Fiocco and Putter, 2011), and is credited to Kragh Andersen et al. (1993). It was collected for a clinical trial on histologically verified liver cirrhosis patients, which took place in Copenhagen from 1962 until 1974. Of a total number of 488 patients, 251 received a hormonal prednisone treatment and 237 received a placebo treatment.

The multi-state model illustrated in Figure 4.1 is appropriate for this data set. Each patient follows a trajectory between three health states: Healthy (1), Sick (2) or Death (3), where state 1 and 2 are transient, and 3 is absorbent. From both transient states, it is possible to transfer to either of the two remaining states. Each permitted transition is indexed by $j = \{1, \ldots, 4\}$ according to the list in Table 4.1. This multi-state model is known in the literature as an illness-death model with recovery (Kragh Andersen et al., 1993).

Table 4.1: State transitions of the multi-state model in Figure 4.1.

1. Healthy (1) → Sick (2)
2. Sick (2) → Healthy (1)
3. Healthy (1) → Death (3)
4. Sick (2) → Death (3)

The time scale of the MS-TTE data in the liver cirrhosis data set is

continuous, and at the time of entry into the study, t_0, each patient is in either of the two transient states *healthy* or *sick*. The data set contains 196 right-censored patients who were still alive at the end of the study, or who withdrew from the study at an earlier stage. Each patient has a set of three covariates that are time-invariant, $\boldsymbol{x} = (x_1, x_2, x_3)$, where $x_1 \in \{0, 1\}$ is the *treatment group*, $x_2 \in \mathbb{R}^+$ is the *starting age* at t_0 and $x_3 \in \{0, 1\}$ refers to the sex of the patient and indicates if the patient is *female*.

4.1.1 Long Format and Data Structure

The liver cirrhosis data is presented on long format, where each row represents a single transition, and an observation may consist of multiple rows (Wreede, Fiocco and Putter, 2011). All rows of the first 5 patients in the data set are displayed in Table A.1. As we can see, the columns *treatment group*, *starting age*, *female* and *starting state* are time-invariant and the same for all rows with the same *patient id*. *Starting state* is included as a separate column for easy access due to the requirements of Algorithm 9 and Algorithm 10. Each row is a transition from a state with index *from* and to a state with index *to*. The transition itself has index *trans*, which is dependent on the columns *from* and *to*, following Table 4.1. The process enters state *to* at time T_{start}, and transitions at time T_{stop}. The column *status* refers to if a transition is censored or not, and *years* is the amount of time spent in the *to* state before a transition takes place, meaning that it is dependent on T_{start} and T_{stop}.

Since there are two possible transitions from each transient state, each uncensored transition is paired by a censored transition. This is consistent with the long format representation we discussed in Chapter 3. In the first row of patient 2, there is a transition from state $2 \to 1$, which makes the competing event of transitioning from $2 \to 3$ censored. The final two rows of a patient can both be censored, which is the case for patient 3, meaning that the the patient is right-censored.

In the notation from Chapter 3, we had that a single observation i consisted of three vectors of the same length, $\boldsymbol{t}_i$, $\boldsymbol{s}_i$ and $\boldsymbol{d}_i$, which contained the transition or censoring times, the transition indices and censor indications respectively, and in addition a vector of covariates $\boldsymbol{x}_i$. The vectors $\boldsymbol{t}_i$, $\boldsymbol{s}_i$ and $\boldsymbol{d}_i$ are equivalent to the columns T_{stop}, *trans* and *status*, and the time-invariant variables of *starting age*, *female* and *treatment group* are equivalent to the covariate vector $\boldsymbol{x}$.

4.2 Naive Approaches to Synthesis of MS-TTE Data

4.2.1 Tabular Synthesis

The next step is to create synthetic MS-TTE data based on the real liver cirrhosis data. Since the synthetic data is on table form, as we have seen in Table A.1, we will first attempt a naive approach. We treat the data set as tabular data and use a tabular synthesiser, which we discussed in Section 2.2.1.

We use CTGAN (L. Xu and Veeramachaneni, 2018) through the Python library *Synthcity* (Qian, Cebere and M. v. d. Schaar, 2023), and generate 20 synthetic rows using 100 epochs and otherwise standard hyperparameter settings. The result is displayed in Table A.2, and we have sorted the rows by *patient id* and T_{start}.

Clearly, these results do not oblige to the required structure of MS-TTE data. This is because tabular synthesisers treat each row as independent. In Table A.2, there is only one row per patient, except for *patient id* 561. This illustrates how this synthesis method fails. The rows of patient 561 begins with an uncensored transition to the absorbent state 3, which should be the final uncensored transition, but this is not the case. Moreover, the T_{start} and T_{stop} times and state transitions do not align across rows and there is no censored competing transition per uncensored transition. The time-invariant covariates also vary across rows of the same patient.

We expected that a tabular synthesiser could recognise the dependencies between the columns *trans*, *from* and *to*, and between T_{start}, T_{stop} and *years*, because the dependencies also hold when we look at each row separately. The majority of the transitions (16 out of 20) are correctly labelled, and the data does not contain any illegal transitions. Nevertheless, the dependencies between the time columns are not recognised, but this could be improved upon by increasing the number of epochs. By using *synthpop* (Nowok, Raab and Dibben, 2016) instead of CTGAN we could assign rules to the dependent variables, as discussed in Section 2.2.1. Alternatively, we could remove the columns *years*, *from* and *to* from the synthesis, and add them as a post-processing step instead.

Because the limitations to the tabular synthesis approach are evident simply by looking at the data in Table A.2, there is no point in evaluating the utility and privacy of the synthetic data further. We move on to the next attempt.

4.2.2 Sequential Synthesis

The tabular synthesis approach clearly failed because the synthesiser did not learn the dependencies across rows. We attempt to solve this by using a sequential synthesiser, which is a more reasonable choice, since each data point consists of several sequences $(\boldsymbol{t}, \boldsymbol{s}, \boldsymbol{d})$. We use the CPAR model (K. Zhang, Patki and Veeramachaneni, 2022) discussed in Section 2.2.2. The column *patient id* is selected as a sequence key, which identifies the rows that belong to the same sequence, and T_{stop} is chosen as a timestamp column, which is called a sequence index[1] in CPAR. The time-invariant variables are labelled as context columns, which secures that they do not vary across rows. We use the standard hyperparameter settings.

The first 5 sequences of the synthetic data set generated by this model are displayed in Table A.3, and we observe that the time-invariant covariates are consistent for all patients. Unfortunately, this is where the similarities ends. The model fails to generate synthetic patients with the required correlations across rows, and suffer from the same issues as the data in Table A.2 in terms of structure of the censoring and consistency of the time variables and transitions. In addition, the data displays illegal transitions, such as transitions from and to the same state, and the *trans* and *female* columns appear to suffer from mode collapse. As we have discussed, this can be a problem for GANs.

Some of these problems, such as having a competing censored transition per uncensored transition and aligning the dependent columns, could be corrected

[1] Preferably, T_{start} should have been labeled as a timestamp column as well, but CPAR currently only allows for one sequence index.

in post-processing steps. However, this would not resolve that the CPAR model clearly has not learnt the structure of the multi-state model.

4.3 Synthesising MSTTE Data Using Weibull Regression

In the previous section, we demonstrated how two off-the-shelf tabular and sequential synthesisers are inadequate for generating MS-TTE data. This motivates the need of a synthesiser tailored to this data type, which is conditioned on a specified multi-state model. In Section 3.6, we proposed Algorithm 11 for this purpose. This synthesiser $\mathcal{S}$ consists of a tabular synthesiser $\mathcal{S}_{tab}$ and an MS-TTE synthesiser $\mathcal{S}_{ms}$, where the latter either assumes that the data follow a clock-forward or a clock-reset model. We will explore both synthesisers $\mathcal{S}^{(cr)}$ and $\mathcal{S}^{(cf)}$ with a Weibull regression MS-TTE model using the liver cirrhosis data set.

In Chapter 2 we described how privacy and utility evaluation methods may require two separate training and tests sets of real data. We use a 80/20 random split to obtain a training set $\mathcal{D}_r^{(train)}$ and test set $\mathcal{D}_r^{(test)}$. Unless otherwise stated, the synthesisers are fitted to $\mathcal{D}_r^{(train)}$, which contains 391 patients.

4.3.1 Using *synthpop* for Tabular Synthesis

As a tabular synthesiser $\mathcal{S}_{tab}$, we choose *synthpop* (Nowok, Raab and Dibben, 2016), which we covered in Section 2.2.1. Among the covariates $\boldsymbol{x}$ and the starting state s_0, we choose the visiting sequence x_2, x_3, x_1, s_0. Furthermore, we use the default option where the first variable in the sequence is sampled with replacement from $\mathcal{D}_r^{(train)}$, and the remaining variables are selected from classification trees fitted to the conditional distributions, as described in Section 2.2.1. We do not adjust the conditioning, meaning that all variables are conditioned on the preceding variables in the visiting sequence.

Age is the only continuous variable, and none of the patients in $\mathcal{D}_r^{(train)}$ has the exact same age. Because we sample with replacement from unique values, this means that each synthetic patient will have an age that is unique for a single patient in $\mathcal{D}_r^{(train)}$, which seems unwise from a privacy point of view[2]. However, it is plausible that such oversights can occur, because we have simply used the default option in *synthpop*. Our objective is not to create a perfectly private synthesiser, but rather to explore and demonstrate evaluation methods. By keeping possible errors in the synthesiser that may manifest themselves as privacy errors downstream, it becomes more compelling to evaluate the privacy of the synthetic data.

4.3.2 Synthesising Processes from MS-TTE Models Using WPHR

Next, we need to generate the state trajectories $\boldsymbol{s}$ and transition times $\boldsymbol{t}$ with an MS-TTE synthesiser $\mathcal{S}_{ms}$, which are on short format. As shown in Algorithm 11, this can be converted to long format in a post processing step, so that the synthetic data is on the same format as in Table A.1.

[2]As a safer choice, we could sample synthetic values from a fitted parametric distribution, such as the normal distribution.

We will use Algorithm 9 for a clock-reset synthesis and Algorithm 10 for a clock-forward process. As described in Algorithm 11, we condition on the output from $\mathcal{S}_{tab}$.

Fitting the Weibull proportional hazards regression (WPHR) models

Both Algorithm 9 and Algorithm 10 require a set of parameters $\boldsymbol{\theta}_j$ for each legal transition $j \in \{1, \ldots, 4\}$, and we fit a WPHR model for each transition. As we recall from Section 3.2.3, a WPHR model has the parameters $\boldsymbol{\theta} = (\lambda, \boldsymbol{\beta_0}, \boldsymbol{\beta})$. From Equation (3.9) we have that the shape parameter σ_j of the Weibull distributions of each transition j is defined as

$$\sigma_j = \exp\left(\beta_{j,0} + \beta_{j,1}x_1 + \beta_{j,2}x_2 + \beta_{j,3}x_3\right),$$

for a patient with covariates $\boldsymbol{x} = (x_1, x_2, x_3)$.

Fitting a clock-reset synthesiser

If we wish to use a clock-reset synthesiser $\mathcal{S}^{(cr)}$, we need to fit the parameters in Algorithm 9 according to a clock-reset model. The *starting age* covariate x_2 indicates the age at the beginning of the process, and is time-invariant. However, as we discussed in Section 3.5.1, age can be used as a semi-time-dependent covariate in a clock-reset model. Therefore, we use x_2', which is age at the last transition. Considering the format of the data displayed in Table A.1, x_2' is obtained from *starting age* + *years*. We use *flexsurv* (Jackson, 2016) to find the maximum likelihood estimates of the parameters for each WPHR clock-reset model and use $\boldsymbol{x}' = \left(x_1, x_2', x_3\right)$ as covariates. This completes line 2 of Algorithm 11.

Fitting a clock-forward synthesiser

The clock-forwars synthesiser $\mathcal{S}^{(cf)}$ requires parameters fitted to a clock-forward model, as discussed in Section 3.6. Then, we will not use semi-time-dependent covariates. Instead we fit the WPHR models using the standard, time-independent covariates $\boldsymbol{x} = (x_1, x_2, x_3)$. Here, we will also use *flexsurv* (Jackson, 2016) to find the maximum likelihood estimates of the parameters of the clock-forward model.

4.3.3 Clock-Reset and Clock-Forward Synthetic Data Sets

By using Algorithm 11 on $\mathcal{D}_r$ as described above, we fit two synthesisers $\mathcal{S}^{(cr)}$ and $\mathcal{S}^{(cf)}$, They are used to generate two synthetic data sets of size $n = 391$, equal in size to $\mathcal{D}_r^{(train)}$. We refer to them as $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$. See Appendix B for details on the implementations in R.

4.4 Utility Evaluation of Synthetic MS-TTE Data

In this section, we demonstrate the utility evaluation methods discussed in Section 2.4 by comparing the real training data $\mathcal{D}_r$ to the two synthetic data sets $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$.

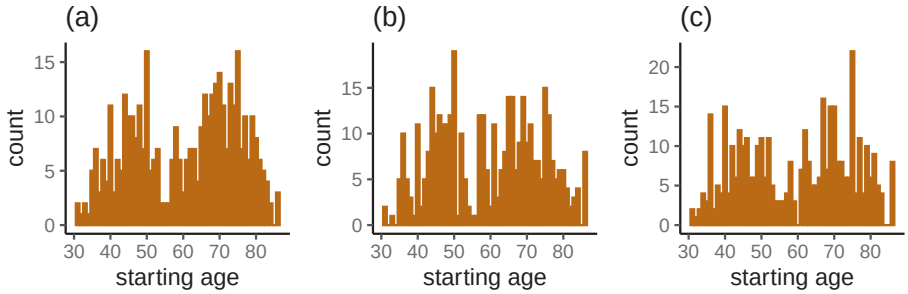


Figure 4.2: Histogram of the *starting age* variable in the data sets (a) $\mathcal{D}_r^{(train)}$, (b) $\mathcal{D}_s^{(cf)}$ and (c) $\mathcal{D}_s^{(cr)}$.

4.4.1 General Utility Evaluation

We begin with a general utility evaluation, where we compare the distributions of the data. Because the general utility metrics described in Section 2.4.1 are tailored to tabular data, we will begin by only considering the time-invariant variables $(\boldsymbol{x}, s_0)$, that is *treatment group*, *starting age*, *female* and *starting state*, generated by $\mathcal{S}_{tab}$. This synthesiser is used to generate these variables for both synthetic data sets $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$, and any difference in performance between the two is due to the randomisation of the synthesiser. We choose to include the evaluation of both data sets to show how synthetic data sets from the same synthesiser can differ, and so that we more clearly can analyse the full performance of each set later on.

Compare marginal distributions

First, we will consider the marginal distributions of x_2, *starting age*, which is continuous. The histograms of the variable for each of the the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$ are displayed in Figure 4.2. We see that the data sets have the same minimum and maximum values, and that the multi-modality of the training data is captured by both synthetic data sets. Recall that the synthetic data points are simply sampled with replacement from the training data, so this is expected.

Next, we will examine the frequency tables of the categorical variables *treatment group*, *female* and *starting state*, which are displayed in Table 4.2. The frequencies appear to be similar, except that the synthetic data sets have a more even distribution of the variables *treatment group* and *female*.

Table 4.2: The frequencies of the categorical variables *treatment group*, *female* and *starting state* of the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$.

	Treatment group		Female		Starting state	
	0	1	0	1	1	2
$\mathcal{D}_r^{(train)}$	187	204	181	210	171	220
$\mathcal{D}_s^{(cf)}$	201	190	184	207	168	223
$\mathcal{D}_s^{(cr)}$	196	195	196	195	176	215

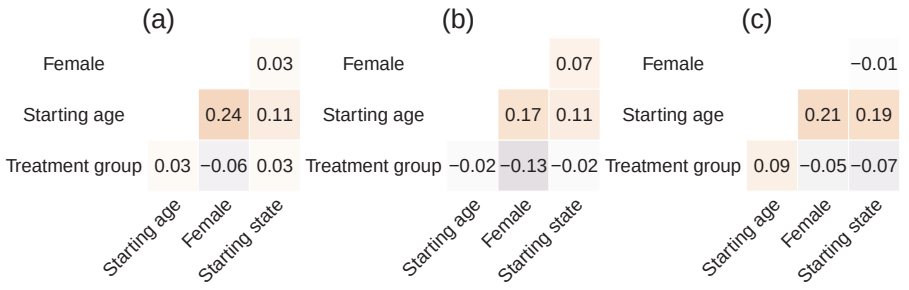


Figure 4.3: Spearman correlation among the variables $(\boldsymbol{x}, s_0)$ of the data sets $\mathcal{D}_r^{(train)}$ (a), $\mathcal{D}_s^{(cf)}$ (b) and $\mathcal{D}_s^{(cr)}$ (c).

As discussed in Section 2.4.1, looking at the data can be a good indicator, but we should also perform pairwise comparisons of the samples through hypothesis tests. Beginning with the continuous variable *starting age*, we use a two-sided KS test, and a chi-square goodness-of-fit test is used for each of the the categorical variables. The p-values of the tests are displayed in Table 4.3. We observe that none of the test indicate a significant difference if we use a significance level of $\alpha = 0.05$, which indicates that $\mathcal{S}_{tab}$ is able to capture the marginal distributions.

Table 4.3: The p-values of hypothesis tests comparing the marginal distributions between $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cf)}$, and between $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cr)}$, with respect to the time-invariant variables $\boldsymbol{x}, s_0$.

	Treatment group	Starting age	Female	Starting state
$\mathcal{D}_s^{(cf)}$	0.16	0.27	0.76	0.76
$\mathcal{D}_s^{(cr)}$	0.36	0.80	0.13	0.61

Correlation

In Section 2.4.1, we discussed how the mean difference in pairwise correlations can be used to evaluate if the correlation structures in the real data are maintained in the synthetic data. Figure 4.3 displays three correlation plots of $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$ respectively. The Spearman and Pearson correlations are very similar. Therefore, we only display the Spearman correlation here. The plots show that the correlation structure is similar between $\mathcal{D}_r^{(train)}$ and the two synthetic data sets. The mean difference in pairwise correlations between $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cf)}$ was 0.0357, and the mean correlation between $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cr)}$ was 0.0070, which is low. In comparison, the mean difference in pairwise correlations between $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_r^{(test)}$ is 0.0351 This indicates that $\mathcal{S}_{tab}$ has captured the correlation structures well.

4.4.2 Specific Utility Evaluation

As we continue to the specific utility evaluation, we will focus on the synthetic data sets $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$ as a whole and not only the time-invariant variables.

Because this includes the state transition processes, we can use $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$ to evaluate the differences between the synthesisers $\mathcal{S}^{(cr)}$ and $\mathcal{S}^{(cf)}$.

As we covered in Section 2.4.2, the focus of the specific utility evaluation is to evaluate the synthetic data with respect to specific tasks. This depends upon the data type, and for our MS-TTE data we will use methods that are specific to survival data.

Fitting survival regression models

Such tasks include using the synthetic data to fit survival regression models. We will fit Weibull regression and Cox regression models, with both the clock-forward and clock-reset approach, to the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, and see how the coefficients of the models fitted to the synthetic data compares to the coefficients of the real model, which we define as the gold standard. Because the synthetic data is generated from a Weibull regression model, we explore if the synthetic data perform better at fitting a Weibull regression model than a Cox regression model. Moreover, we investigate if $\mathcal{D}_s^{(cr)}$ are better at fitting clock-reset models than clock-forward models, and conversely for $\mathcal{D}_s^{(cf)}$. As we discussed in Section 2.1.3, research shows that synthetic data sets generated from synthesisers with deep learning structures tend to perform better on models that are similar in structure to their synthesisers. We will investigate if this holds for our synthesisers as well.

We begin with the Weibull regression models, and we will fit both clock-reset and clock-forward models, in the same manner as we described in Chapter 3, and we use all covariates $\boldsymbol{x}$. This results in the parameters $\lambda_j, \beta_{j,0}, \beta_{j,1}, \beta_{j,2}, \beta_{j,3}$ for each transition $j \in \{1, \ldots, 4\}$. The parameters of the clock-reset model are listed in Table 4.4, and the clock-forward parameters are listed in Table 4.5. The parameters estimated from $\mathcal{D}_r^{(train)}$ in Table 4.4 are the same as those used to synthesise $\mathcal{D}_s^{(cr)}$ in Algorithm 9. Likewise, the parameters estimated from $\mathcal{D}_r^{(train)}$ in Table 4.5 were used to synthesise $\mathcal{D}_s^{(cf)}$ in Algorithm 10.

From Table 4.4, we observe that the parameters of $\mathcal{D}_s^{(cr)}$ were 11 out of 20 times closest to the gold standard parameters of $\mathcal{D}_r^{(train)}$ compared to $\mathcal{D}_s^{(cf)}$. However, in the cases where the parameters of $\mathcal{D}_s^{(cf)}$ were closer, the parameters were close to 0 or there were small differences between the performances of the synthetic data sets. We should also consider the overlaps in confidence intervals, and we will use the interval overlap utility (IOU) score from Definition 2.4.3. The results are displayed in Table 4.6. Recall that a score of 1 is a perfect overlap, while a negative score indicates no overlap. Interestingly, the highest scores are overlaps between $\mathcal{D}_r$ and $\mathcal{D}_s^{(cf)}$. However, the overall performance is better by $\mathcal{D}_s^{(cr)}$, as there are no negative scores. The median IOU scores of $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$ are 0.85 and 0.80 respectively.

A clock-forward model is more complex, and we see in Table 4.5 that the $\mathcal{D}_s^{(cf)}$ are closest to the gold standard parameters 18 out of 20 times, and the only times. Looking at the IOU scores in Table 4.7, it becomes clear that the $\mathcal{D}_s^{(cr)}$ struggled to fit a clock-forward model, as many of the confidence intervals do not overlap. The median IOU score is 0.84 for $\mathcal{D}_s^{(cf)}$, and 0.36 for $\mathcal{D}_s^{(cr)}$.

This shows that both synthetic data sets perform best at fitting models of similar structure as their synthesiser. However, $\mathcal{D}_s^{(cf)}$ had a more stable

Table 4.4: MLEs of the parameters of a **clock-reset** Weibull regression model fitted to the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, for each of the $j \in \{1, \ldots, 4\}$ transitions.

j	$\mathcal{D}$	λ	β_0	β_1	β_2	β_3
1	$\mathcal{D}_r^{(train)}$	1.0417	0.0068	-0.1574	0.0658	0.1260
	$\mathcal{D}_s^{(cf)}$	0.9786	0.0082	-0.3100	0.0628	0.0677
	$\mathcal{D}_s^{(cr)}$	1.0604	0.0047	-0.1199	0.0692	0.3306
2	$\mathcal{D}_r^{(train)}$	1.0692	0.0014	0.0900	0.0713	0.2764
	$\mathcal{D}_s^{(cf)}$	1.2020	0.0013	-0.0424	0.0706	0.4418
	$\mathcal{D}_s^{(cr)}$	1.1365	0.0005	-0.2101	0.0874	0.3438
3	$\mathcal{D}_r^{(train)}$	1.0516	0.4261	0.3169	0.0027	0.1545
	$\mathcal{D}_s^{(cf)}$	0.8821	0.4819	0.2141	-0.0014	0.3028
	$\mathcal{D}_s^{(cr)}$	1.0480	0.5330	0.2968	-0.0007	0.4464
4	$\mathcal{D}_r^{(train)}$	0.6658	0.0061	0.1138	0.0662	0.1614
	$\mathcal{D}_s^{(cf)}$	1.0899	0.0023	0.0824	0.0782	0.4023
	$\mathcal{D}_s^{(cr)}$	0.6844	0.0093	0.1454	0.0606	0.2569

Table 4.5: MLEs of the parameters of a **clock-forward** Weibull regression model fitted to the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, for each of the $j \in \{1, \ldots, 4\}$ transitions.

j	$\mathcal{D}$	λ	β_0	β_1	β_2	β_3
1	$\mathcal{D}_r^{(train)}$	0.9951	0.0091	-0.1774	0.0631	0.0975
	$\mathcal{D}_s^{(cf)}$	1.0476	0.0070	-0.2806	0.0677	0.0240
	$\mathcal{D}_s^{(cr)}$	1.1257	0.0064	-0.0487	0.0726	0.0771
2	$\mathcal{D}_r^{(train)}$	1.6101	0.0001	0.1788	0.0966	0.3808
	$\mathcal{D}_s^{(cf)}$	1.5725	0.0002	0.0001	0.0906	0.4607
	$\mathcal{D}_s^{(cr)}$	1.2490	0.0059	-0.1139	0.0667	0.2780
3	$\mathcal{D}_r^{(train)}$	0.8321	0.7473	0.2788	-0.0025	0.1091
	$\mathcal{D}_s^{(cf)}$	0.8010	0.6263	0.2416	-0.0002	0.1902
	$\mathcal{D}_s^{(cr)}$	1.0316	0.3040	0.2914	0.0147	0.2798
4	$\mathcal{D}_r^{(train)}$	1.2741	0.0010	0.2172	0.0904	0.2885
	$\mathcal{D}_s^{(cf)}$	1.4066	0.0009	0.2116	0.0898	0.3199
	$\mathcal{D}_s^{(cr)}$	0.9639	0.3777	0.2692	0.0214	-0.0397

performance than $\mathcal{D}_s^{(cr)}$. This strengthens the claim that more complex synthesisers have a higher utility than synthesisers with a simpler structure, as $\mathcal{D}_s^{(cf)}$ were more versatile than $\mathcal{D}_s^{(cr)}$.

Next, we turn to the Cox regression models, where we also fit both clock-reset and clock-forward variants. The coefficients of the clock-reset model are found in Table 4.8, and we observe that the $\mathcal{D}_s^{(cr)}$ coefficients were closest to the gold standard coefficients of $\mathcal{D}_r^{(train)}$ 5 out of 12 times compared to $\mathcal{D}_s^{(cf)}$. In Table 4.9, we list the coefficients of the clock-forward model. Here, the $\mathcal{D}_s^{(cf)}$ coefficients were closest to the gold standard 9 out of 12 times compared to

Table 4.6: IOU scores of the parameters in Table 4.4. Measures the overlap between the 95% confidence intervals of $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cf)}$, and $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cr)}$, respectively.

j	$\mathcal{D}$	λ	β_0	β_1	β_2	β_3
1	$\mathcal{D}_s^{(cf)}$	0.68	0.91	0.69	0.87	0.89
	$\mathcal{D}_s^{(cr)}$	0.85	0.70	0.86	0.86	0.57
2	$\mathcal{D}_s^{(cf)}$	0.65	0.81	0.85	0.88	0.81
	$\mathcal{D}_s^{(cr)}$	0.84	0.54	0.63	0.58	0.88
3	$\mathcal{D}_s^{(cf)}$	0.02	0.95	0.79	0.82	0.69
	$\mathcal{D}_s^{(cr)}$	0.90	0.89	0.90	0.86	0.37
4	$\mathcal{D}_s^{(cf)}$	-1.17	0.47	0.91	0.57	0.60
	$\mathcal{D}_s^{(cr)}$	0.88	0.82	0.89	0.80	0.85

Table 4.7: IOU scores of the parameters in Table 4.5. Measures the overlap between the 95% confidence intervals of $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cf)}$, and $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cr)}$, respectively.

j	$\mathcal{D}$	λ	β_0	β_1	β_2	β_3
1	$\mathcal{D}_s^{(cf)}$	0.78	0.79	0.79	0.80	0.86
	$\mathcal{D}_s^{(cr)}$	-0.20	0.81	0.66	0.89	0.86
2	$\mathcal{D}_s^{(cf)}$	0.86	0.90	0.79	0.86	0.85
	$\mathcal{D}_s^{(cr)}$	0.12	-0.36	0.86	0.28	0.87
3	$\mathcal{D}_s^{(cf)}$	0.79	0.83	0.93	0.89	0.83
	$\mathcal{D}_s^{(cr)}$	-0.57	-0.07	0.62	-0.27	0.71
4	$\mathcal{D}_s^{(cf)}$	0.53	0.83	0.90	0.88	0.90
	$\mathcal{D}_s^{(cr)}$	-1.45	-42.49	0.43	-3.64	0.63

$\mathcal{D}_s^{(cr)}$. The IOU scores of both models are listed in Table 4.10. The median IOU scores of the clock-reset model is 0.743 and 0.739 for $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$ respectively, meaning that $\mathcal{D}_s^{(cr)}$ performed slightly better than $\mathcal{D}_s^{(cf)}$. This is a worse performance than for the clock-reset Weibull regression task, which we would expect because the synthesisers do not follow a Cox regression model. As for the clock-forward model, the median IOU scores were 0.581 for $\mathcal{D}_s^{(cr)}$ and 0.841 for $\mathcal{D}_s^{(cf)}$. This surprising result shows that $\mathcal{D}_s^{(cf)}$ performed as well on this task as for the clock-forward Weibull model, which indicates high utility. It is also surprising that $\mathcal{D}_s^{(cr)}$ performed better here compared to the performance of $\mathcal{D}_s^{(cr)}$ at the clock-forward Weibull regression model.

Furthermore, we may use the confidence intervals to evaluate how the synthetic data can be used for model selection, which we discussed in Section 2.1.3. We illustrate by considering the 95% confidence intervals of β_1, the *treatment group* coefficients, and β_3, the *female* coefficients, of transition $j = 1$ displayed in Table 4.4. The confidence intervals of these coefficients are shown in Table 4.11. Say we wished to test the significance of β_1 and β_3 for the models trained on $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, and test the hypotheses $H_0 : \beta_k = 0$ for $k \in \{1, 3\}$. Then, with a confidence level of $\alpha = 0.05$ we would accept H_0 for

Table 4.8: MLEs of the parameters of a **clock-reset** Cox regression model fitted to the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, for each of the $j \in \{1, \ldots, 4\}$ transitions.

j	$\mathcal{D}$	β_1	β_2	β_3
1	$\mathcal{D}_r^{(train)}$	-0.1496	0.0554	0.0834
	$\mathcal{D}_s^{(cf)}$	-0.2857	0.0630	0.0924
	$\mathcal{D}_s^{(cr)}$	-0.1222	0.0700	0.3380
2	$\mathcal{D}_r^{(train)}$	0.0834	0.1127	0.4007
	$\mathcal{D}_s^{(cf)}$	-0.0510	0.0731	0.4261
	$\mathcal{D}_s^{(cr)}$	-0.2218	0.0887	0.3432
3	$\mathcal{D}_r^{(train)}$	0.3059	-0.0030	0.1528
	$\mathcal{D}_s^{(cf)}$	0.1886	-0.0026	0.2881
	$\mathcal{D}_s^{(cr)}$	0.2958	-0.0009	0.4275
4	$\mathcal{D}_r^{(train)}$	0.1902	0.0959	0.2466
	$\mathcal{D}_s^{(cf)}$	0.1033	0.0803	0.4163
	$\mathcal{D}_s^{(cr)}$	0.1423	0.0604	0.2348

Table 4.9: MLEs of the parameters of a **clock-forward** Cox regression model fitted to the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, for each of the $j \in \{1, \ldots, 4\}$ transitions.

j	$\mathcal{D}$	β_1	β_2	β_3
1	$\mathcal{D}_r^{(train)}$	-0.2311	0.0581	0.0748
	$\mathcal{D}_s^{(cf)}$	-0.2663	0.0711	0.0445
	$\mathcal{D}_s^{(cr)}$	-0.1239	0.0731	0.3667
2	$\mathcal{D}_r^{(train)}$	0.1708	0.0998	0.3734
	$\mathcal{D}_s^{(cf)}$	-0.0131	0.0898	0.4628
	$\mathcal{D}_s^{(cr)}$	-0.2156	0.0868	0.3207
3	$\mathcal{D}_r^{(train)}$	0.3381	-0.0097	0.1044
	$\mathcal{D}_s^{(cf)}$	0.2599	-0.0025	0.1561
	$\mathcal{D}_s^{(cr)}$	0.2638	0.0011	0.4000
4	$\mathcal{D}_r^{(train)}$	0.2420	0.0987	0.2351
	$\mathcal{D}_s^{(cf)}$	0.2830	0.0886	0.3378
	$\mathcal{D}_s^{(cr)}$	0.2824	0.0628	0.4629

both $k \in \{1, 3\}$ using the model trained on $\mathcal{D}_r^{(train)}$, while the model trained on $\mathcal{D}_s^{(cf)}$ would accept H_0 for β_3, but find that β_1 is significant. Oppositely, the model trained on $\mathcal{D}_s^{(cr)}$ would find β_1 significant and not β_3. Then, none of the synthetic data sets were able to select the same model as $\mathcal{D}_r^{(train)}$. In a scenario such as Example 2.1.6, it is less critical that the synthetic models found more significant variables than $\mathcal{D}_r^{(train)}$, because this can be resolved by performing a final variable selection on real data.

j	$\mathcal{D}$	β_1	β_2	β_3
1	$\mathcal{D}_s^{(cf)}$	0.72	0.67	0.91
	$\mathcal{D}_s^{(cr)}$	0.86	0.35	0.46
2	$\mathcal{D}_s^{(cf)}$	0.85	0.08	0.85
	$\mathcal{D}_s^{(cr)}$	0.62	0.48	0.87
3	$\mathcal{D}_s^{(cf)}$	0.75	0.91	0.72
	$\mathcal{D}_s^{(cr)}$	0.90	0.87	0.41
4	$\mathcal{D}_s^{(cf)}$	0.86	0.51	0.72
	$\mathcal{D}_s^{(cr)}$	0.89	-0.20	0.88

(a) Clock-reset.

j	$\mathcal{D}$	β_1	β_2	β_3
1	$\mathcal{D}_s^{(cf)}$	0.91	0.44	0.91
	$\mathcal{D}_s^{(cr)}$	0.78	0.32	0.39
2	$\mathcal{D}_s^{(cf)}$	0.79	0.77	0.85
	$\mathcal{D}_s^{(cr)}$	0.53	0.70	0.88
3	$\mathcal{D}_s^{(cf)}$	0.84	0.67	0.90
	$\mathcal{D}_s^{(cr)}$	0.85	0.48	0.37
4	$\mathcal{D}_s^{(cf)}$	0.91	0.69	0.84
	$\mathcal{D}_s^{(cr)}$	0.89	-0.17	0.64

(b) Clock-forward.

Table 4.10: IOU scores of the parameters in Table 4.8 are displayed in (a) and Table 4.9 are displayed in (b). They measure the overlap between the 95% confidence intervals of $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cf)}$, and $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_s^{(cr)}$, respectively.

Table 4.11: 95% confidence intervals of β_1 and β_3 displayed in Table 4.4 for the transition $j = 1$.

j	$\mathcal{D}$	$CI\,(\beta_1)$		$CI\,(\beta_3)$	
1	$\mathcal{D}_r^{(train)}$	(-0.421,	0.106)	(-0.144,	0.400)
	$\mathcal{D}_s^{(cf)}$	(-0.525,	-0.095)	(-0.150,	0.285)
	$\mathcal{D}_s^{(cr)}$	(-0.310,	0.070)	(0.137,	0.525)

Comparing Kaplan-Meier curves

Like Guillaudeux et al. (2023) and Norcliffe et al. (2023), we compare of Kaplan-Meier curves to evaluate the utility of $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_s^{(cf)}$. They are non-parametric survival curves which can be used for censored data (Kleinbaum and Klein, 2005). We can compare the Kaplan-Meier curves by performing a log-rank test, which is a form of a Chi-squared test (see Definition 2.4.1), where the failure times are divided into categories. H_0 is that the survival function of the synthetic and real data are the same. Note that these curves display the survival trends of the data sets as a whole, regardless of covariate values.

In Figure 4.4 we plot the Kaplan-Meier curves of $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_r^{(train)}$ for all transitions. The most striking observation is that in *(a)* and *(b)*, the survival curves continue much longer for $\mathcal{D}_s^{(cr)}$ than $\mathcal{D}_r^{(train)}$. This is because the study that the liver cirrhosis data is collected from only lasted 12 years, which means that many patients are censored. The synthetic survival curves also utilise censored observations, because of the structure of the competing risks-model, as discussed in Section 3.4. The difference is that the process of each synthetic patient continues until the absorbent state is reached. The Weibull distribution can be long-tailed, and we have not included an age or time limit in the model. This results in some unrealistic survival times. However, this is not captured by the log-rank tests of transitions 1 and 2, because there is no support for real data after $t = 12$. With a significance level of $\alpha = 0.05$, none of the tests are rejected. The same conclusion holds for the differences in survival curves between $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$, which are displayed in Figure 4.5. The validity of this

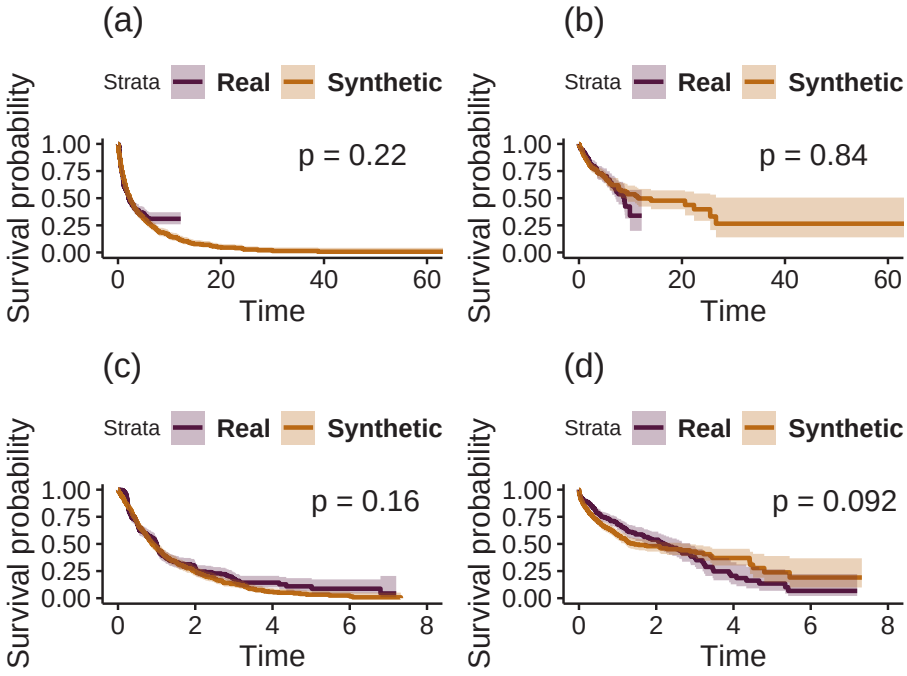


Figure 4.4: Kaplan-Meier survival curves with 95% confidence intervals and corresponding log-rank test p-values of $\mathcal{D}_s^{(cr)}$ and $\mathcal{D}_r^{(train)}$ for each transition $j \in \{1, \ldots, 4\}$ in the figures *(a)-(d)*.

conclusion is disputable, because it is recommended that the compared curves should have a similar censoring pattern, which is not the case here (Therneau, 2023).

Suggestions for further specific utility evaluation

There are many other potential tasks that we could use to evaluate synthetic MS-TTE data. By conducting more utility evaluation tasks, we will increase the credibility of the final conclusion of the utility evaluation. However, it is not feasible to test every possible task, and due to practicality, we need to constrain the length of this section. That being said, we wish to suggests some options for further utility evaluation for MS-TTE data.

Classification tasks which are discussed in Section 2.4.2 can also be performed on MS-TTE data. We can for example classify the treatment group or sex of a patient based on the other covariates and possibly also semi-time-dependent covariates that describe the state transition process, such as time spent in state 1, number of transitions and the time of death if it has occurred. Then, the TSTR and TRTR framework described in Algorithm 1 can be used.

Moreover, we can use Algorithm 1 to compare how well competing risks models trained on synthetic data predict the hazards of unseen patients in a test set $\mathcal{D}_r^{(test)}$ at specified time points. We can compare the predicted and observed risk, using the Concordance-index or Brier score (Z. Zhang et al., 2018).

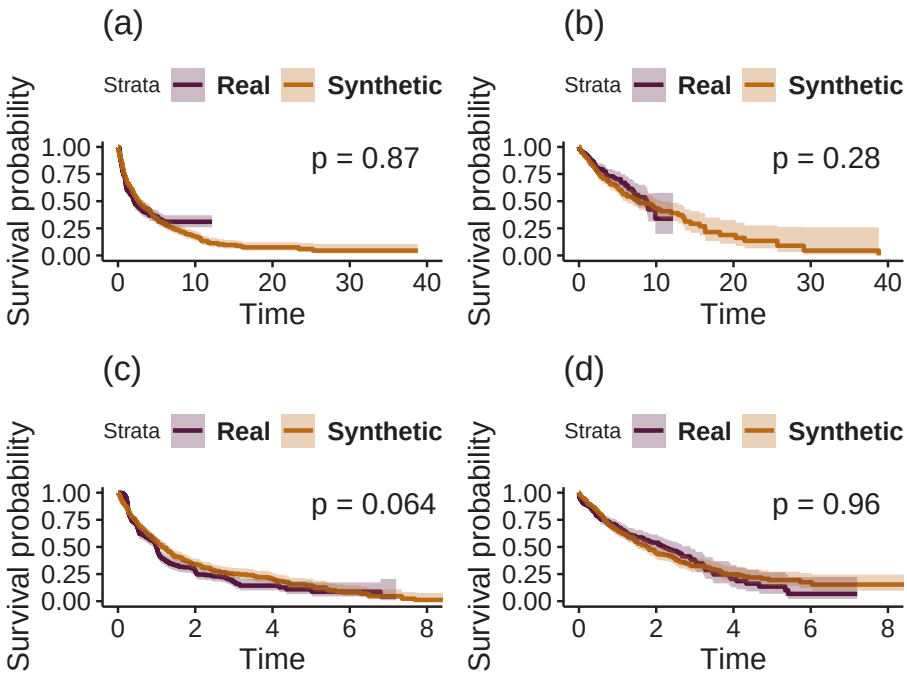


Figure 4.5: Kaplan-Meier survival curves with 95%confidence intervals and corresponding log-rank test p-values of $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_r^{(train)}$ for each transition $j \in \{1, \ldots, 4\}$ in the figures *(a)-(d)*.

Unless we add covariates that describe the transition history, none of the methods we have considered so far are able to challenge the validity of the Markov assumption of the synthetic data, which is a consequence of the structure of our synthesisers. In the next section, we will use the four distance evaluation procedures defined in Section 2.6 to consider the compete state transition history of the data.

4.5 Privacy and Utility Evaluation of MS-TTE Data Using Distance Metrics

In Section 2.6, we described how distance metrics can be used to evaluate the privacy and utility of synthesisers, and we will now apply this to MS-TTE data. The DCR and NNDR metrics are based on the distances between single data points, which is not easily defined for MS-TTE data. Where tabular data consists of points with the same set of variables, MS-TTE data points contain several vectors, and the vectors of different points have different lengths. In this section we aim to find a technique to define distances between MS-TTE data points. Then, we will use the distance evaluation procedures from Section 2.6 to evaluate the utility and privacy of the clock-reset synthesiser $\mathcal{S}^{(cr)}$ and the clock-forward synthesiser $\mathcal{S}^{(cf)}$.

4.5.1 Removing Censored Observations

In this section, we will simplify the problem by only considering uncensored real data. When we remove all the censored observations from the Liver Cirrhosis Data, we get a data set with 292 patients. As explained in Section 2.6, the different procedures require different splits between train and test data. The reduced data set is still on long format and we display the first patients in this data set $\mathcal{D}_r$ in Table A.4. A real training data set $\mathcal{D}_r^{(train)}$ is used to train the synthesisers $\mathcal{S}^{(cr)}$ and $\mathcal{S}^{(cf)}$ as before.

4.5.2 Distance Between Uncensored MS-TTE Data Points

As we have established, an uncensored data point can be represented using the vectors $\boldsymbol{t}, \boldsymbol{s}, \boldsymbol{x}$, where the first two have the same length and they are both on short format. We aim to find a more compact representation of a single data point, as this will make it easier to define the distance between two points.

In a first attempt, we utilise that we only have two transient states and one absorbent state in our state transition model, as seen in Figure 4.1. Thus, we can represent the data of each patient using only two vectors. In the first vector we will include the starting state as an additional variable in addition to the covariates age, sex and treatment group, as we did when we generated the tabular data. This gives us the vector (x_1, x_2, x_3, s_0). Next, the rest of the state transition trajectory vector $\boldsymbol{s}$ can be found simply by knowing the length of the transition times vector $\boldsymbol{t}$. This is because we know the starting state, that the final state is absorbent and the length of $\boldsymbol{s}$. Then, we also know the intermediate states, as the process alternates between the two transient states until the final state is reached.

Example 4.5.1 [Representing each patient as two vectors]

We show how the data of the patient with *patient id* 2 in Table A.4 can be represented using two vectors. The vector with the covariates starting age, sex and treatment group and the additional variable starting state becomes $(\boldsymbol{x}, s_0) = (47.93, 1, 0, 2)$, and the vector with the transition times is $\boldsymbol{t} = (0.00, 0.69, 1.19, 2.20, 4.75, 5.72, 6.76)$. The information on transitions is encoded in starting state, and we do not need additional information.

Each point can be represented by two vectors, and we explore the option of measuring the distance between each point by considering the two vectors separately. This raises several questions, one of which is how to weigh the importance of the separate distances, and secondly, how to determine distance measures for both data types.

Comparing the distances between two observations $\left(x_{i,1}, x_{i,2}, x_{i,3}, s_{i,0}\right)$ and $\left(x_{j,1}, x_{j,2}, x_{j,3}, s_{j,0}\right)$ is straightforward, because they always have the same length. We can then consider the distance between points in a four dimensional space, and use a metric like L1 or the Euclidean distance.

It is a greater challenge to find the distances between two vectors $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$, because they do not necessarily have the same length, and comparing vectors of different lengths is not straightforward (Bozkaya, Yazdani and Özsoyoğlu, 1997; Sun et al., 2011). Additionally, this representation does not generalise to

models with more than two transient states, and we aspire to create a method that can be used by other multi-state models, like for instance the model in Figure 3.2. To avoid the drawbacks of this representation, we explore another option.

By discretising the transition times, we can represent the information contained in $\boldsymbol{s}$ and $\boldsymbol{t}$ as a single sequence. So far, we have assumed that the liver cirrhosis data is continuous with years as the time unit, but we now discretise the time using days as the step size. We can use sequences where each element represents one day, starting from when the patient was admitted into the study and until death, and each element contains the state the patient was in at the given day. We still assume that the distribution of t is continuous, as in Chapter 3, but we round to the closest day. A similar discretising can be done for any continuous data by selecting an appropriate step size.

This representation still consists of sequences of differing lengths, and we present a method which can make them equally sized, as this will simplify the comparison process. Since the last element of every sequence is 3, which signals a death, we can grow the shortest sequence with values of 3, which can be interpreted as that the patient remains in state 3 as time passes.

Example 4.5.2 [Preparing for sequence comparison]

Say we have a sequence $A = [1, 1, 1, 1, 1, 2, 2, 3]$. A remained in the starting state 1 for 5 days, then spent 2 days in State 2 before they reached the absorbing state 3 on day 8. We want to compare A to another shorter sequence $B = [1, 1, 2, 1, 3]$, and we expand it to $B' = [1, 1, 2, 1, 3, 3, 3, 3]$. We can then compare A and B' using the normalised Hamming distance.

The normalised Hamming distance is a simple method for comparing two sequences of the same length, and we will use it to find the distance between two state sequences. It finds the average $0-1$ Loss of the sequences (M. M. Deza and E. Deza, 2014). We display it in Algorithm 12.

Algorithm 12 Normalised Hamming distance

Input:

$\boldsymbol{x}, \boldsymbol{y} \leftarrow$ sequences of length $n \in \mathbb{N}$

Output:

$\|\boldsymbol{x} - \boldsymbol{y}\|_H \in [0, 1]$

1:

$$\|\boldsymbol{x} - \boldsymbol{y}\|_H = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left(x_i \neq y_i\right)$$

Example 4.5.3 [Find the normalised Hamming distance]

We want to find the normalised Hamming distance between A and B' from

Example 4.5.2.

$$A = [1, 1, 1, 1, 1, 2, 2, 3]$$
$$B' = [1, 1, 2, 1, 3, 3, 3, 3]$$

Since the elements with indices $1, 2, 4, 8$ are the same, and $n = 8$ we find that the distance is $\|A - B'\|_H = 0.5$.

We proceed with the normalised Hamming distance. Nonetheless, we wish to point out some possible limitations to this distance representation. The choice of representation is a question of what we consider as close. For example, we might think that a patient with a short survival time should be close to patients with longer survival times if the state transition processes are similar up until the first patient's death. Picture a synthesiser that creates synthetic processes that are copies of the first half of the processes of real data. Because only the first half of the real process is the same, this similarity will not be noticed by the normalised Hamming distance, which will return a distance of 0.5. These synthetic patients can be used to identify real training patients, which is something that we want to avoid. An alternative that solves this issue could be to also shrink the longest sequence to the size of the shortest, and then find the normalised Hamming distance as before. This method will disregard the differences in survival times, which arguably should be important for the distance. However, it could be a useful as a supplementary measure.

Example 4.5.4 [Distant patients]

Sequence $A = [1, 1, 2, 1, 1, 2, 2, 3]$ and $B' = [1, 1, 2, 3, 3, 3, 3, 3]$ from Example 4.5.3 have a distance of 0.5, even though the beginning of sequence A is close to sequence $B = [1, 1, 2, 3]$. The normalised Hamming distance between B and an adjusted A of the same length, $A' = [1, 1, 2, 1]$ is $\|B - A'\|_H = 0.25$.

Another result of this representation is that patients with many transitions can be close to patients with few transitions, which may not be intended.

Example 4.5.5 [Close patients]

Sequence $A = [1, 1, 1, 1, 1, 2, 2, 3]$ only has 2 transitions, sequence $C = [1, 2, 1, 2, 1, 2, 2, 3]$ has 6 transitions. Still, the normalised Hamming distance between A and C is only $\|A - C\|_H = 0.25$.

We return to the distance between vectors $(\boldsymbol{x}, s_0)$. By using the discretised sequence representation of the state trajectories, we do not need to include s_0, as it is represented in the first value of the sequence. Now, the difference between two covariate vectors is defined as the distance between two points in a three-dimensional space. Sex and treatment group are binary and age is continuous. We standardise all the covariates as shown in Algorithm 13. For each covariate, we subtract $\hat{\mu}$ and divide by $\hat{\sigma}$ to ensure that each covariate contribute equality to the distance. The estimates are based on the training set

$\mathcal{D}_r^{(train)}$. We then find the Euclidean distance between the normalised covariate vectors. Note that if some of the covariates are considered as more sensitive than others, then we can adjust Algorithm 13 so that proximity between the sensitive covariates holds more weight.

Algorithm 13 Standardise a covariate vector

Input:
$\boldsymbol{x} \leftarrow$ covariate vector
$\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}$ estimated means and standard deviations for each covariate
Output:
$\boldsymbol{x'} \leftarrow$ standardised $\boldsymbol{x}$
for $x_i \in \boldsymbol{x}$ **do**
  $x_i' \leftarrow (x_i - \hat{\mu}_i) \,/\, (\hat{\sigma}_i)$
end for

We now have a procedure for measuring the distance between pairs of both state trajectories and the covariate vectors. Our task is now to combine them into a joint distance metric, which we use to find the distance between any two uncensored MS-TTE data points. The contributions of the covariates and state trajectories are regulated using a weight w. Algorithm 14 describes the process of finding the closest distance to another patient among a set of candidates.

Algorithm 14 Find the closest patient

Input:
$\boldsymbol{x}, \boldsymbol{A} \leftarrow$ covariates and sequence of a patient
$\left\{(\boldsymbol{y_1}, \boldsymbol{B_1}), \ldots, (\boldsymbol{y_n}, \boldsymbol{B_n})\right\} \leftarrow$ a set of n patients with covariates and sequences
$w \leftarrow$ weight
Output:
$d \in [0, 1]$
$\boldsymbol{x'} \leftarrow$ Algorithm 13($\boldsymbol{x}$)
$L \leftarrow$ empty list
for $\boldsymbol{y}, \boldsymbol{B} \in \left\{(\boldsymbol{y_1}, \boldsymbol{B_1}), \ldots, (\boldsymbol{y_n}, \boldsymbol{B_n})\right\}$ **do**
  $\boldsymbol{y'} \leftarrow$ Algorithm 13($\boldsymbol{y}$)
  $covs_d \leftarrow \text{Euclidean}\left(\boldsymbol{x'}, \boldsymbol{y'}\right)$
  $\boldsymbol{A'}, \boldsymbol{B'} \leftarrow$ grow $(\boldsymbol{A}, \boldsymbol{B})$ {Grow the shortest sequence}
  $seq_d \leftarrow \|\boldsymbol{A'} - \boldsymbol{B'}\|_H$
  $L \leftarrow$ append $\left(w \cdot cov_d + (1 - w)\, seq_d\right)$
end for
$d \leftarrow \min(L)$

4.5.3 DCR and NNDR experiment

We use Algorithm 14 to find DCR and NNDR metrics and evaluate both the utility and privacy following the evaluation procedures described in Section 2.6. Both the synthetic clock forward data $\mathcal{D}_s^{cf}$ and synthetic time reset data $\mathcal{D}_s^{cr}$ are evaluated. We use the α values $0.05, 0.5$ for both DCR and NNDR, as this measures both privacy and utility. In total, we have four distance

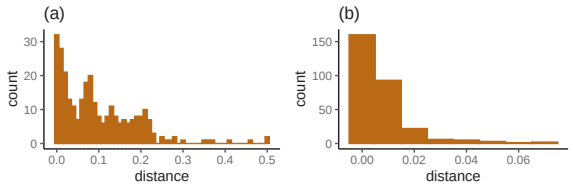


Figure 4.6: Histogram comparing the intra-data set DCR distances of $\mathcal{D}_r$ with different weights. In (a), $w = 0$ so that only the sequences contribute to the distance. In (b), $w = 1$ so that only the covariates contribute to the distance.

metrics for each of the four procedures. As stated in Section 2.6, we use the Wilcoxon rank-sum test to evaluate the difference in distances. The test assumes independence between the two samples of distances or between the sample and baseline (Devore, Berk and Carlton, 2021). For all procedures we use $k = 20$ splits.

Tuning the weight hyper-parameter

First, we need to determine the weight parameter w in Algorithm 14. The Hamming distance is bounded between $[0, 1]$, while the standardised covariate distance is not bounded. We wish that both distances should contribute within the same magnitude. There is no given way to find such a weight, and our suggestion is to compare the intra-data set DCR distances (Definition 2.6.6) of each point in $\mathcal{D}_r$ with the weights $w = 0$ and $w = 1$. We plot them in Figure 4.6 and we see that both are skewed towards the right, but the magnitude of the covariate distances is lower than the magnitude of the sequence distances. In Figure 4.6 (a), we have that the mean value is 0.0966, while (b) has a mean of 0.00779. We wish to increase the contribution of the covariates, and as $0.0966/0.00779 \approx 12$ we can set the weight to $1 - 1/(12 + 1) \approx 0.92$. To illustrate why this is, say that we have two values $x = 1$ and $y = 12$, and we wish that they should contribute equally to a weighted sum. With a weight $w = 1 - 1/13$ we get that

$$w \cdot x + (1 - w) \cdot y = \left(1 - \frac{1}{13}\right) \cdot 1 + \left(1 - \left(1 - \frac{1}{13}\right)\right) \cdot 12 = \frac{12}{13} + \frac{12}{13}.$$

Proceeding with $w = 0.92$, we plot the intra-data set DCR and NNDR distances for $\mathcal{D}_r$ in Figure 4.7 to illustrate the distribution of the distances. In procedure 1 and 2, we will show how this weight differs from the default weight $w = 0.5$.

Procedure 1

We begin with procedure 1, which evaluates both the privacy and utility as described in Algorithm 6. The procedure uses 80/20 train-test set splits of the full set of 292 uncensored patients, so that both of the synthetic data sets and the test set contain 92 patients. This makes procedure 1 the fastest to fit, because we do not consider as many distances.

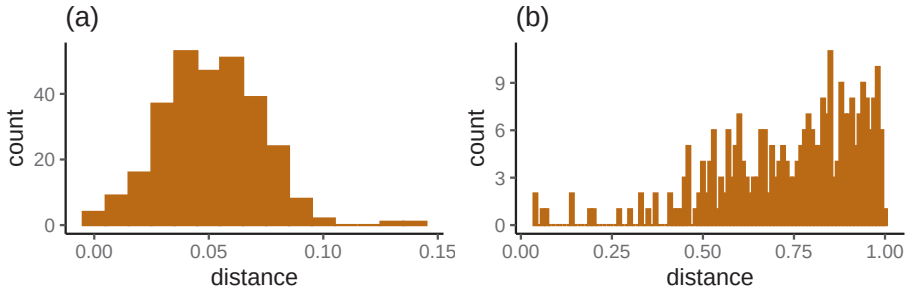


Figure 4.7: Histogram of the intra-data set distances of $\mathcal{D}_r$ with $w = 0.92$. In (a), we display the DCR distances, and in (b) we display the NNDR distances.

First, we show the results of using Algorithm 14 with $w = 0.5$ as the single-point distance metric. We display the output of Algorithm 6 for the two different synthesisers $\mathcal{S}^{(cf)}$ and $\mathcal{S}^{(cr)}$ in Table 4.12. Each column represents a specific distance metric. The first row is the distance $\|\mathcal{D}_r^{(test)} - \mathcal{D}_r^{(train)}\|$ with respect to each metric, and this is a constant value because we have a single training and test set. The second and third rows are the mean distances from the $K = 20$ stochastic distances. The last rows are the p-values of a Wilcoxon signed rank test which tests if the random sample of distances $\|\mathcal{D}_s^k - \mathcal{D}_r^{(train)}\|$, where $k \in \{1, \ldots, 20\}$, are similarly distributed as $\|\mathcal{D}_r^{(test)} - \mathcal{D}_r^{(train)}\|$.

The first column, which is the $\alpha = 0.05$ percentile of the DCR distances, indicate that the synthetic data points that are closest to the training data have a similar distance pattern to unseen test data, which is an indicator of good privacy. This is the case both for the $\mathcal{D}_s^{(cr)}$ and the $\mathcal{D}_s^{(cf)}$ data sets. Next, we see from the second column that this is also the case for the $\alpha = 0.5$ percentile of the DCR distances, which indicates sufficient general utility. The synthetic data sets are neither too close nor too far from the real data. The NNDR metrics are mainly a measure of how well the outliers in $\mathcal{D}_r^{(train)}$ are protected. Low values indicate that the synthetic data is much closer to a single data point in $\mathcal{D}_r^{(train)}$ than the second closest point, which would cause a privacy concern. This is not the case here. Overall, we see no difference in the performance of the two synthesisers, and the tests indicate both satisfactory privacy and utility.

Table 4.12: DCR and NNDR distances and p-values following from procedure 1 using $w = 0.5$.

	DCR		NNDR	
α	0.05	0.5	0.05	0.5
$\|\mathcal{D}_s^k - \mathcal{D}_r^{(train)}\|$	0.0542	0.1703	0.3402	0.8212
$\|\mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(train)}\|$	0.0465	0.1629	0.3168	0.7959
$\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(train)}\|$	0.0533	0.2009	0.3737	0.8249
p-value $\mathcal{D}_s^{(cf)}$	0.3810	0.5714	0.5714	0.5714
p-value $\mathcal{D}_s^{(cr)}$	1.0000	0.0952	0.7619	1.0000

When we adjust the weight in Algorithm 14 to $w = 0.92$, so that both the

distances between covariates and the distances between sequences contribute within the same magnitude to the total distance, we get the result displayed in Table 4.13. With a significance level of 0.05, there is still no significant difference between the distances between the synthetic data sets and $\mathcal{D}_r^{(train)}$ and the distance between $\mathcal{D}_r^{(test)}$ and $\mathcal{D}_r^{(train)}$, and the same conclusion holds.

Table 4.13: DCR and NNDR distances and p-values following from procedure 1 using $w = 0.92$.

	DCR		NNDR	
α	0.05	0.5	0.05	0.5
$\|\mathcal{D}_s^k - \mathcal{D}_r^{(train)}\|$	0.0216	0.0632	0.4420	0.8081
$\overline{\|\mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(train)}\|}$	0.0183	0.0508	0.3671	0.7941
$\overline{\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(train)}\|}$	0.0199	0.0585	0.3720	0.8346
p-value $\mathcal{D}_s^{(cf)}$	0.6667	0.0952	0.3810	0.9524
p-value $\mathcal{D}_s^{(cr)}$	0.8571	0.3810	0.6667	0.4762

Procedure 2

Algorithm 7 describes how the second procedure is carried out for a given distance metric. It uses one-sided tests, which is because we use it to evaluate the privacy of the synthetic data sets. We are interested to see if the synthetic data is closer to its training data than unseen data. As described in 2.6, we now need 50/50 train-test set splits of the 292 uncensored patients.

The returned values from Algorithm 7 for each distance metric are displayed in Table 4.14, using weight $w = 0.5$. With a significance level of 0.05, all null hypotheses, which states that the synthetic data is closer or equally close to the test data compared to the training data, are accepted. This is the case for the outliers, as we can see when $\alpha = 0.05$, and the the overall data sets, judging by the median value. The conclusion holds for both $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$ and for all DCR and NNDR measures.

Table 4.14: DCR and NNDR distances and p-values following from procedure 2 with weights $w = 0.5$.

	DCR		NNDR	
α	0.05	0.5	0.05	0.5
$\overline{\|\mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(train)}\|}$	0.0534	0.2032	0.3415	0.8164
$\overline{\|\mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(test)}\|}$	0.0522	0.1977	0.3267	0.7931
$\overline{\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(train)}\|}$	0.0635	0.2280	0.3550	0.8340
$\overline{\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(test)}\|}$	0.0625	0.2333	0.3497	0.8291
p-value $\mathcal{D}_s^{(cf)}$	0.4733	0.9460	0.7355	0.9948
p-value $\mathcal{D}_s^{(cr)}$	0.6510	0.1105	0.6510	0.7355

When we readjust the weight so that the closeness in covariates contributes to the total distance, we get a different result. Because we now get a result where the majority of the tests are significant, using a significance level of 0.05, we choose to correct the p-values using the Benjamini-Hochberg (Ferreira and Zwinderman, 2006) method to control for the false discovery rate of repeated tests. The results with the adjusted p-values are displayed in Table 4.15. We see that for all the distance metrics except $\alpha = 0.5$ NNDR, both the $\mathcal{D}_s^{(cf)}$ and the $\mathcal{D}_s^{(cr)}$ data sets are significantly closer to the training set than to unseen test data. This indicates that the synthesisers $\mathcal{S}^{(cf)}$ and $\mathcal{S}^{(cr)}$ are overfit to the training data. The synthetic patients have a higher proximity to real patients in the training set than what we expect from random proximity. As a result, the synthetic data sets can potentially be used to target $\mathcal{D}_r^{(train)}$ in MIAs. The DCR distances show that this is the case for both the patients with the closest distance, but also the overall synthetic data set. The significance level is reduced when we turn to the NNDR distances. Using $\alpha = 0.05$, we still see that the patients in $\mathcal{D}_s^{(cf)}$ and $\mathcal{D}_s^{(cr)}$ with the lowest NNDR scores are too close to a single patient in $\mathcal{D}_r^{(train)}$ compared to the second nearest patient. This indicates that the outliers in $\mathcal{D}_r^{(train)}$ are at risk. The distance metric $\alpha = 0.5$ NNDR show that $\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(train)}\|$ is significantly closer than $\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(test)}\|$, indicating that the patients in $\mathcal{D}_s^{(cr)}$ in general are too close to a single point in $\mathcal{D}_r^{(train)}$ compared to the second closest. There is no significant result for $\mathcal{D}_s^{(cf)}$.

Table 4.15: DCR and NNDR distances and adjusted p-values following from procedure 2 with weights $w = 0.92$.

	DCR		NNDR	
α	0.05	0.5	0.05	0.5
$\overline{\|\mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(train)}\|}$	0.0194	0.0587	0.3620	0.8109
$\overline{\|\mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(test)}\|}$	0.0269	0.0661	0.4153	0.8214
$\overline{\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(train)}\|}$	0.0229	0.0629	0.3858	0.8317
$\overline{\|\mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(test)}\|}$	0.0295	0.0719	0.4505	0.8353
p-value $\mathcal{D}_s^{(cf)}$	2.34E-08	<1E-08	0.0063	0.0319
p-value $\mathcal{D}_s^{(cr)}$	0.0001	<1E-08	0.0059	0.3198

The difference between Table 4.14 and Table 4.15 is explained by that the proximity between covariates in the two synthetic and $\mathcal{D}_r^{(train)}$ data sets cause a privacy violating proximity, while the synthetic sequences are not too close to the sequences in $\mathcal{D}_r^{(train)}$. This is not surprising as the $\mathcal{S}_{tab}$ using *synthpop* (Nowok, Raab and Dibben, 2016) is more complex than the $\mathcal{S}_{ms}^{(cf)}$ and $\mathcal{S}_{ms}^{(cr)}$ synthesisers that generate the sequences. Because of their structures, the transitions of each patient are generated without regarding the full transition history.

Procedure 3 and 4

Lastly, we perform procedure 3 and 4 by following Algorithm 8. Recall that procedure 3 measures both the privacy and utility, and that procedure 4 measures the utility. As these procedures do not use a test set, we use the full data set of 292 patients as $\mathcal{D}_r^{(train)}$, and the synthetic data sets are of the same size. Because the size of the data sets are large we do not correct for the difference in size of intra- and inter-data set distances as discussed in Section 2.6. The large size of the data sets results in that these procedures have a higher computational cost.

We now only consider the balanced weight $w = 0.92$, and we display the results of both procedures in Table 4.16. As none of the p-values show significance with a significance level of 0.05, we have not corrected the p-values for repeated testing.

Table 4.16: DCR and NNDR distances and adjusted p-values following from procedure 3 and 4 with weights $w = 0.92$.

	DCR		NNDR	
α	0.05	0.5	0.05	0.5
$\overline{\lVert \mathcal{D}_s^{(cf)} - \mathcal{D}_r^{(train)} \rVert}$	0.0156	0.0475	0.3335	0.8122
$\overline{\lVert \mathcal{D}_s^{(cr)} - \mathcal{D}_r^{(train)} \rVert}$	0.0174	0.0534	0.3753	0.8433
$\overline{IDD\left(\mathcal{D}_r^{(train)}\right)}$	0.0144	0.0516	0.3189	0.8143
$\overline{IDD\left(\mathcal{D}_s^{(cf)}\right)}$	0.0143	0.0485	0.3358	0.8027
$\overline{IDD\left(\mathcal{D}_s^{(cr)}\right)}$	0.0160	0.0541	0.3122	0.8228
p-value $\mathcal{D}_s^{(cf)}$ test 3	0.4088	0.1371	0.5087	0.7411
p-value $\mathcal{D}_s^{(cr)}$ test 3	0.2857	0.4762	0.1905	0.0952
p-value $\mathcal{D}_s^{(cf)}$ test 4	1.0000	0.1905	0.7619	0.3810
p-value $\mathcal{D}_s^{(cr)}$ test 4	0.7619	0.2857	0.9524	0.5714

We first consider the results of procedure 3. As discussed in Section 2.6, this procedure is similar in structure and interpretation as procedure 1. It tests if the synthetic data sets are closer or more distant to their $\mathcal{D}_r^{(train)}$ than a proximity between points that is expected in the real data. Here we use the intra-data set distance of $\mathcal{D}_r^{(train)}$ as the baseline. With a significance level of 0.05, none of the differences in the distances are significant. This indicates high utility and privacy, and we draw the same conclusions as for procedure 1 using $w = 0.92$. Note that the distances in Table 4.13 and 4.16 are not comparable, because we have different sample sizes.

Procedure 4 performs a utility evaluation, and measures the heterogeneity of $\mathcal{D}_s^{cf}$ and $\mathcal{D}_s^{cr}$ compared to $\mathcal{D}_r^{(train)}$, which we discussed in Section 2.4. Also here, we do not find a significant difference between the intra-data set distances of $\mathcal{D}_s^{cf}$ and $\mathcal{D}_s^{cr}$ compared to $\mathcal{D}_r^{(train)}$. This indicates that the synthetic data sets are not prone to mode-collapse, and that they are as well spread out in the domain space as the real data. We see that this is the case both for the data sets as a whole, because we reach the same conclusions using $\alpha = 0.05$ and

$\alpha = 0.5$. This holds for both $\mathcal{D}_s^{cf}$ and $\mathcal{D}_s^{cr}$ and for both the DCR and NNDR metrics.

Discussion

We have now completed a total of 32 tests, which may be too excessive. As discussed, the structure of procedure 1 and 3 are similar, and we can remove one of them from our test framework. The 3rd procedure has the benefit of not requiring a test set, which means that the sample sizes were larger. However, this requires a higher computational cost. Procedure 2 is especially useful to evaluate the privacy of the synthetic data sets. We see that the potentially privacy violating proximity that we exposed in this test was not discovered by procedure 1 and 3, despite that they also measure the privacy. Furthermore, procedure 4 is also important, because we can evaluate the heterogeneity of the synthetic data.

Overall, the synthetic data score high on the utility tests, and lower on the privacy tests. We see no overall difference in performance between $\mathcal{S}^{cf}$ than $\mathcal{S}^{cr}$, which indicates that the added complexity of the time inhomogeneity in $\mathcal{S}^{cf}$ does not contribute to a higher utility for the liver cirrhosis data. Note that this is different from the conclusion reached in Section 4.4.2, when we found that $\mathcal{D}_s^{(cf)}$ was more versatile than $\mathcal{D}_s^{(cr)}$ for fitting Weibull and Cox regression models.

Moreover, we have shown the importance of tuning the weight between the covariate distance and the trajectory distance. A weakness to this evaluation framework is that it only considers uncensored data. By finding a distance metric which allows for a comparison of censored and uncensored data points, this can extended to censored data as well.

4.6 Membership Inference Attack Experiment

Successful membership inference attacks (MIAs) have the potential of obstructing the goal of synthetic data, namely protecting real data. As we discussed in Section 2.5.2, these types of attacks attempt to take advantage of overfitted synthesisers. In this section, we will perform an MIA attack on a synthetic clock-reset data set, which we in this section simply refer to as $\mathcal{D}_s$, using the methods discussed previously.

As discussed, an unsuccessful attack does not mean that the synthetic data is immune to these types of attacks. With non-differentially private synthetic data, we have no guarantee against the precision of another attack with a different design or with increased computational capacity. Even though an unsuccessful attack cannot lead to any conclusions, a successful attack explicitly tells us that the privacy of the synthetic data is lacking.

This constructed attack is designed as a worst-case scenario. We, the adversary in this case, are familiar with the structure of the synthesiser, but we do not have access to its fitted parameters. As a second advantage, we have access to a real data set that closely resembles the target data. This is done by dividing the full data set $\mathcal{D}_r$ into three folds. Half is used as the training set $\mathcal{D}_r^{(train)}$, 10 points form the data set $\mathcal{D}_0^{cand}$ and are used as candidates for

the attack, and the remaining points form a data set $\mathcal{D}_r^{MIA}$ which we use to perform the MIA.

The data set $\mathcal{D}_r^{(train)}$ is the target data for the attack. We aim to train a classifier that given a synthetic data set $\mathcal{D}_s$ can predict if a specific candidate patient x is in the training data of the unknown synthesiser $\mathcal{S}$, given that $\mathcal{D}_s$ is generated from $\mathcal{S}$. We use the data set $\mathcal{D}_r^{MIA}$ to train such a classifier $\mathcal{C}_x$ by following Algorithm 2. This is an ideal situation from an adversary's point of view, since $\mathcal{D}_r^{MIA}$ comes from the same source as $\mathcal{D}_r^{(train)}$ and we can assume that they follow the same distribution.

In our attack, we have 20 candidate patients that we suspect are in $\mathcal{D}_r^{(train)}$. The first 10 candidates are from the data set $\mathcal{D}_0^{cand}$, and they are disjoint from both $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_r^{MIA}$. The remaining 10 candidates are randomly sampled from $\mathcal{D}_r^{(train)}$, and they are stored in $\mathcal{D}_1^{cand}$. We wish to train a classifier $\mathcal{C}_{cand}$ for each $cand \in \mathcal{D}_0^{cand}$ and $cand \in \mathcal{D}_1^{cand}$ by using Algorithm 2. If the attack is successful, then the classifiers are able to identify the candidates in $\mathcal{D}_1^{cand}$ as members of $\mathcal{D}_r^{(train)}$ by predicting a score close to 1, while the candidates in $\mathcal{D}_0^{cand}$ are not classified as members of $\mathcal{D}_r^{(train)}$ and should be given scores close to 0. We use a confusion matrix to show how many of the classifications that are correct.

The full attack is described in Algorithm 15. In line 2, we fit a classifier for each patient that is a member of $\mathcal{D}_r^{(train)}$, and in line 5 we do the same for the patients that are not members of $\mathcal{D}_r^{(train)}$. In line 7 we evaluate all of the 20 classifiers by using them to predict membership in the training data of $\mathcal{D}_s$.

Algorithm 15 MIA on 20 suspected patients

Input:
$\mathcal{D}_r^{MIA} \leftarrow$ data set similar to $\mathcal{D}_r^{(train)}$
$K \leftarrow$ amount of shadow synthetic data set pairs
$\mathcal{D}_s \leftarrow$ a synthetic data set generated from $\mathcal{D}_r^{(train)}$
$\mathcal{D}_1^{cand} \leftarrow$ suspected patients in $\mathcal{D}_r^{(train)}$
$\mathcal{D}_0^{cand} \leftarrow$ suspected patients not in $\mathcal{D}_r^{(train)}$
Output:
$cm \leftarrow$ confusion matrix of the attack

1: **for** $cand1_i \in \mathcal{D}_1^{cand}$ **do**
2: $\quad C1_i \leftarrow$ Algorithm 2 $\left(\mathcal{D}_r^{MIA}, K, cand1_i\right)$
3: **end for**
4: **for** $cand0_i \in \mathcal{D}_0^{cand}$ **do**
5: $\quad C0_i \leftarrow$ Algorithm 2 $\left(\mathcal{D}_r^{MIA}, K, cand0_i\right)$
6: **end for**
7: $cm \leftarrow$ Evaluation$(\boldsymbol{C1}, \boldsymbol{C0}, \mathcal{D}_s)$

We use $K = 20$ for all classifiers $\mathcal{C}_{cand}$ so that 20 pairs of bootstrapped data sets and shadow synthesisers are used to fit the classifiers. As we in this case know the exact structure of $\mathcal{S}$, we fit the shadow synthesisers to the shadow data sets in the same way as $\mathcal{S}$ is fitted to $\mathcal{D}_r^{(train)}$. Similar to Kuppa, Aouad and Le-Khac (2021) we will use random forest models to train our classifiers.

In Shokri et al. (2017) neural networks are used, but they use much larger data sets.

As discussed in Section 2.5.2, we fit the classifiers $\mathcal{C}_{cand}$ by labelling the shadow synthetic data sets that has the candidate *cand* in their shadow training set. Ideally, we would like to find a way to represent each data set in a more compact form. If the space of possible synthetic data sets were discrete, with a total size of n, we could represent each data set through one-hot-encoding with a vector of size n. Since there is no limit to how many transitions each patient can make, it is not possible to bound the number of possible synthetic MS-TTE data sets that are on the format displayed in Table A.1. Instead, we choose to add an extra variable to each row in the data set, and do row-based classification. The variable contains a label that indicates if the candidate *cand* was in the training set or not. A weakness to this approach is that the data set is not considered as a whole. If we input a synthetic data set $\mathcal{D}_s$ to a fitted classifier $\mathcal{C}_{cand}$, we then predict, row by row, if the row belongs to a data set that is generated from a synthesiser with *cand* in its training set. The final prediction is made based on an average.

To account for the randomisation of the synthesising process, we use six different synthetic data sets. We also wish to investigate if some patients are easier to classify than others, so we display the results for each patient. In Table 4.17 and Table 4.18 we can see the predictions of the probabilistic classifiers. They predict the probability that each candidate is in $\mathcal{D}_r^{(train)}$. A successful attack should recognise the patients in Table 4.17 as patients that are not used in the training of the synthetic data sets, and classify them as 0, and similarly Table 4.18 should classify the synthetic data sets as 1. While Table 4.18 does have a higher mean than Table 4.17, both have a mean below and close to, 0.5. There are no obvious differences between the synthetic data sets, which indicates that the randomisation process of the synthesiser does not affect the success of the attacks. Some patients, such as *pat6* and *pat7* in Table 4.17 and *pat3* in Table 4.18 seem somewhat more recognisable as patients in or outside $\mathcal{D}_r^{(train)}$ respectively.

Table 4.17: The predictions of the classifiers of the 10 candidates in $\mathcal{D}_0^{cand}$ that are not members of $\mathcal{D}_r^{(train)}$. A successful attack returns predictions close to 0. We use six different synthetic data sets that are all generated from $\mathcal{D}_r^{(train)}$.

	$\mathcal{D}_s^{(1)}$	$\mathcal{D}_s^{(2)}$	$\mathcal{D}_s^{(3)}$	$\mathcal{D}_s^{(4)}$	$\mathcal{D}_s^{(5)}$	$\mathcal{D}_s^{(6)}$	mean
$cand0_1$	0.60	0.45	0.48	0.60	0.47	0.48	0.51
$cand0_2$	0.32	0.55	0.37	0.43	0.44	0.43	0.42
$cand0_3$	0.45	0.51	0.44	0.54	0.60	0.60	0.52
$cand0_4$	0.44	0.41	0.42	0.46	0.40	0.45	0.43
$cand0_5$	0.50	0.44	0.43	0.46	0.49	0.46	0.46
$cand0_6$	0.33	0.39	0.32	0.33	0.29	0.29	0.33
$cand0_7$	0.29	0.39	0.33	0.36	0.34	0.32	0.34
$cand0_8$	0.35	0.40	0.47	0.37	0.38	0.40	0.39
$cand0_9$	0.50	0.43	0.51	0.49	0.52	0.47	0.49
$cand0_{10}$	0.36	0.38	0.39	0.50	0.39	0.46	0.41
mean	0.41	0.43	0.42	0.45	0.43	0.44	0.43

Table 4.18: The predictions of the classifiers of the 10 candidates in $\mathcal{D}_1^{cand}$ that are members of $\mathcal{D}_r^{(train)}$. A successful attack returns predictions close to 1. We use six different synthetic data sets that are all generated from $\mathcal{D}_r^{(train)}$.

	$\mathcal{D}_s^{(1)}$	$\mathcal{D}_s^{(2)}$	$\mathcal{D}_s^{(3)}$	$\mathcal{D}_s^{(4)}$	$\mathcal{D}_s^{(5)}$	$\mathcal{D}_s^{(6)}$	mean
$cand1_1$	0.34	0.45	0.40	0.36	0.39	0.52	0.41
$cand1_2$	0.35	0.43	0.41	0.48	0.44	0.44	0.42
$cand1_3$	0.59	0.64	0.58	0.59	0.56	0.57	0.59
$cand1_4$	0.41	0.45	0.36	0.43	0.48	0.52	0.44
$cand1_5$	0.54	0.58	0.46	0.52	0.43	0.49	0.50
$cand1_6$	0.35	0.36	0.32	0.36	0.42	0.40	0.37
$cand1_7$	0.60	0.47	0.47	0.61	0.53	0.49	0.53
$cand1_8$	0.48	0.47	0.35	0.40	0.44	0.50	0.44
$cand1_9$	0.38	0.44	0.43	0.40	0.45	0.42	0.42
$cand1_{10}$	0.46	0.56	0.49	0.50	0.52	0.57	0.52
mean	0.45	0.48	0.43	0.46	0.46	0.49	0.46

All in all, this is an unsuccessful attack, and from the confusion matrix in Table 4.19, we get that the recall is $R = 0.3$. As discussed, this is not enough to conclude that the synthetic data is secure. We have only performed attacks with 10 patients in $\mathcal{D}_r^{(train)}$ as candidates. Since some patients are easier to classify, we should assume that there are remaining patients in $\mathcal{D}_r^{(train)}$ with a greater privacy risk. The attack could be improved by increasing the number of candidates or by performing a hill-climbing search as described in Algorithm 3. Moreover, the design of the classifiers can surely be improved, which could also result in a more powerful attack.

Table 4.19: Confusion Table.

	Predicted	
Actual	Present	Not present
Present	18	42
Not present	11	49

CHAPTER 5

Differentially Private MS-TTE Data

5.1 Constructing a Differentially Private Synthesiser for MS-TTE Data

This chapter aims to demonstrate how we can generate and evaluate differentially private multi-state time-to-event (MS-TTE) data. To do so, we will use the same liver cirrhosis data as in Chapter 4, which follows the multi-state model illustrated in Figure 4.1. We first introduced differential privacy in Section 2.5.3, and then we proposed differentially private approaches to MS-TTE modelling in Section 3.7. Combined, this makes us equipped to discuss how a fully differentially private synthesiser can be constructed.

Algorithm 11 showed how a compete synthesiser for MS-TTE data $\mathcal{S}$ consists of two synthesisers, a tabular synthesiser, $\mathcal{S}_{tab}$ and an MS-TTE synthesiser, $\mathcal{S}_{ms}$. Both need to be differentially private in order to make $\mathcal{S}$ differentially private. The composition property of differential privacy, which we discussed in Section 2.5.3, is used to control the total privacy budget.

The training set $\mathcal{D}_r^{(train)}$ and test set $\mathcal{D}_r^{(test)}$ of the liver cirrhosis data follow the same split as in Section 4.3, and include censored data, unless stated otherwise.

5.1.1 Differential Privacy of a Tabular Synthesiser

Earlier we have used *synthpop* (Nowok, Raab and Dibben, 2016) to generate the time-invariant variables $\left(\boldsymbol{x}_i, s_{i,0}\right)$ per synthetic patient $i \in \{1, \ldots, n\}$. Because *synthpop* currently does not allow for a differentially private version of the synthesiser we have used, we will instead use DP-GAN (Xie et al., 2018) to synthesise the tabular data. Recall our discussion of GANs from Chapter 2. We use the implementation of the *SynthCity* Python package (Qian, Cebere and M. v. d. Schaar, 2023) to fit a DP-GAN to our training data. This method requires hyper-parameter tuning. First, we need to select the maximum number of training epochs of the generator. The default number of epochs in *SynthCity* is 2000, but we find that the fitting process often converges before 500 epochs for our data set, and that increasing the number of epochs does not improve the results. We keep the other default options, such as structure of the discriminator and generator.

The value of ε regulates the level of privacy, as seen in Definition 2.5.1. We need to find an ε that is as low as possible, while still has sufficient utility. We examine three different values of ε, and we will fit a DP-GAN with each ε five times to control the variation caused by the randomisation process. They are evaluated using the general evaluation framework for tabular data presented in Section 2.4.1, which we demonstrated on non-differentially private synthetic data in Section 4.4.1.

The first four columns of Table 5.1 are parallel to Table 4.3, and the table contains the p-values of tests that check if there is a significant difference between the marginal distributions of the differentially private synthetic data sets and $\mathcal{D}_r^{(train)}$. As before, we use chi-square goodness-of-fit test for the categorical variables and a KS test for the continuous variable *starting age*. For each $\varepsilon \in \{0.5, 1, 5\}$, we have five synthetic data sets, generated from five distinct differentially private synthesisers.

Table 5.1: Adjusted p-values of tests comparing the marginal distributions, and the mean difference in covariance of differentially private $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$, using three different values of ε.

ε	p-values				mean corr
	tr grp	age	female	st state	diff
0.5	2.07E-04	<1E-08	3.70E-03	<1E-08	0.130
	<1E-08	<1E-08	<1E-08	<1E-08	0.105
	<1E-08	<1E-08	<1E-08	<1E-08	0.114
	5.10E-03	<1E-08	<1E-08	1.51E-08	0.124
	<1E-08	9.41E-07	<1E-08	<1E-08	0.092
1	<1E-08	<1E-08	<1E-08	<1E-08	0.196
	<1E-08	<1E-08	<1E-08	<1E-08	0.122
	<1E-08	<1E-08	1.87E-01	8.25E-05	0.082
	<1E-08	<1E-08	1.76E-08	<1E-08	0.135
	1.41E-06	<1E-08	<1E-08	1.31E-01	0.064
5	<1E-08	<1E-08	9.12E-03	1.56E-01	0.095
	<1E-08	<1E-08	<1E-08	<1E-08	0.087
	<1E-08	<1E-08	<1E-08	<1E-08	0.104
	<1E-08	2.92E-07	<1E-08	<1E-08	0.079
	<1E-08	2.41E-08	1.20E-02	1.08E-01	0.058

We see that almost all tests have very low p-values, indicating that there is a significant difference between each $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$. Because we have performed many significant tests, we have adjusted the p-values using the Benjamini-Hochberg method to control for the false discovery rate. While some of the marginal distributions, such as *starting state* get some results where there is no significant difference using a significance level of 0.05, mostly we do not see any improvement when ε is increased, which is unexpected. The difference in mean Spearman covariance slightly improves. The mean values across the five data sets for increasing values of ε are 0.113 for $\varepsilon = 0.5$, 0.112 for $\varepsilon = 1$, and 0.085 for $\varepsilon = 5$. In comparison, the recall that the mean difference in pairwise correlations between $\mathcal{D}_r^{(train)}$ and $\mathcal{D}_r^{(test)}$ is 0.0351. While the difference decreases with increasing values of ε, the overall difference in correlation indicates low utility, and the result is worse than what we saw in Chapter 4.

Recall from Section 2.5.3 that each time we refit an $\varepsilon-$differentially private synthesiser, we spend a privacy budget of ε. This means that we cannot refit synthesisers as we have done here, and handpick the one with the best utility. Instead, they should be randomly selected. We use the first synthetic data set in each category for further analysis, and name them $\mathcal{D}_s^{(0.5)}$, $\mathcal{D}_s^{(1)}$ and $\mathcal{D}_s^{(5)}$.

In Figure 5.1 we plot the *starting age* distributions of $\mathcal{D}_r^{(train)}$ and the three synthetic data sets $\mathcal{D}_s^{(0.5)}$, $\mathcal{D}_s^{(1)}$ and $\mathcal{D}_s^{(5)}$. We see that neither of the differentially private synthetic data sets manage to capture the two modes in $\mathcal{D}_r^{(train)}$, and in $\mathcal{D}_s^{(0.5)}$ and $\mathcal{D}_s^{(1)}$, the minimum observed age is much higher than what we observe in $\mathcal{D}_r^{(train)}$. Note that DP-GAN is not developed for tabular data, which could explain that the multi-modality is lost. See the discussion on GANs tailored to tabular data in Section 2.2.1 for further details.

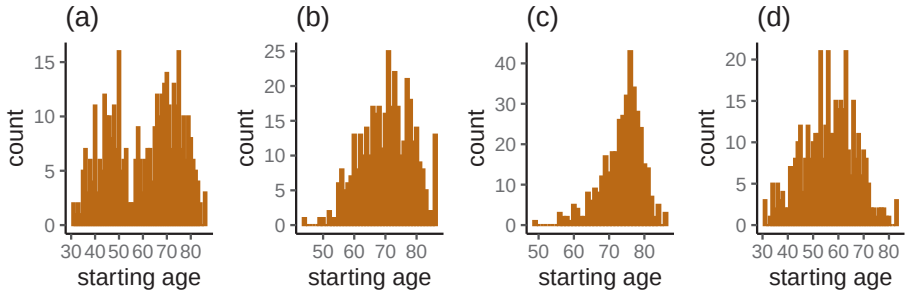


Figure 5.1: Histogram of the *starting age* variable in the data sets (a) $\mathcal{D}_r^{(train)}$, (b) $\mathcal{D}_s^{(0.5)}$, (c) $\mathcal{D}_s^{(1)}$ and (d) $\mathcal{D}_s^{(5)}$.

When we inspect the differences in frequencies in Table 5.2 we also see that the utility is low, confirming the result of the hypothesis tests. In $\mathcal{D}_s^{(1)}$, all the majority categories in $\mathcal{D}_r^{(train)}$ have very few values. The only variable that is close to the frequency in $\mathcal{D}_r^{(train)}$ is *Starting state* in $\mathcal{D}_s^{(1)}$, which is supported by the p-values in Table 5.1.

Table 5.2: The frequencies of the categorical variables *treatment group*, *female* and *starting state* for the data sets $\mathcal{D}_r^{(train)}$, $\mathcal{D}_s^{(0.5)}$, $\mathcal{D}_s^{(1)}$ and $\mathcal{D}_s^{(5)}$.

	Treatment group		Female		Starting state	
	0	1	0	1	1	2
$\mathcal{D}_r^{(train)}$	187	204	181	210	171	220
$\mathcal{D}_s^{(0.5)}$	224	167	152	239	257	134
$\mathcal{D}_s^{(1)}$	377	14	369	22	367	24
$\mathcal{D}_s^{(5)}$	292	99	155	236	185	206

The correlation structure of $\mathcal{D}_r^{(train)}$ in Figure 5.2 (a) is best captured by $\mathcal{D}_s^{(5)}$ in Figure 5.2 (d), while $\mathcal{D}_s^{(1)}$ in Figure 5.2 (b) has the worst resemblance. None of the data sets manage to capture the positive correlation between *starting age* and *female*, like we see in Figure 4.3 that the non-differentially private synthetic data sets did.

We choose to move forward with the data set $\mathcal{D}_s^{(5)}$, which we will use to generate multi-state trajectories. Most of the marginal distributions in this data set do not follow the distributions in $\mathcal{D}_r^{(train)}$, but overall this data set performed best. For example, the minimum and maximum values of *starting age* are correctly captured, and the difference between the marginal distributions of *starting state* are insignificant. However, a value of $\varepsilon = 5$ is a considerable privacy cost, and we discuss this in Section 5.3.

5.1.2 Differential Privacy of a MS-TTE Synthesiser

In Chapter 4, we generated new synthetic processes from Weibull proportional hazards regression (WPHR) models, and we used a model per transition. The parameters $\boldsymbol{\theta}_j = (\lambda_j, \beta_{j,0}, \beta_{j,1}, \beta_{j,2}, \beta_{j,3})$ of each model $j \in \{1, \ldots, 4\}$ were

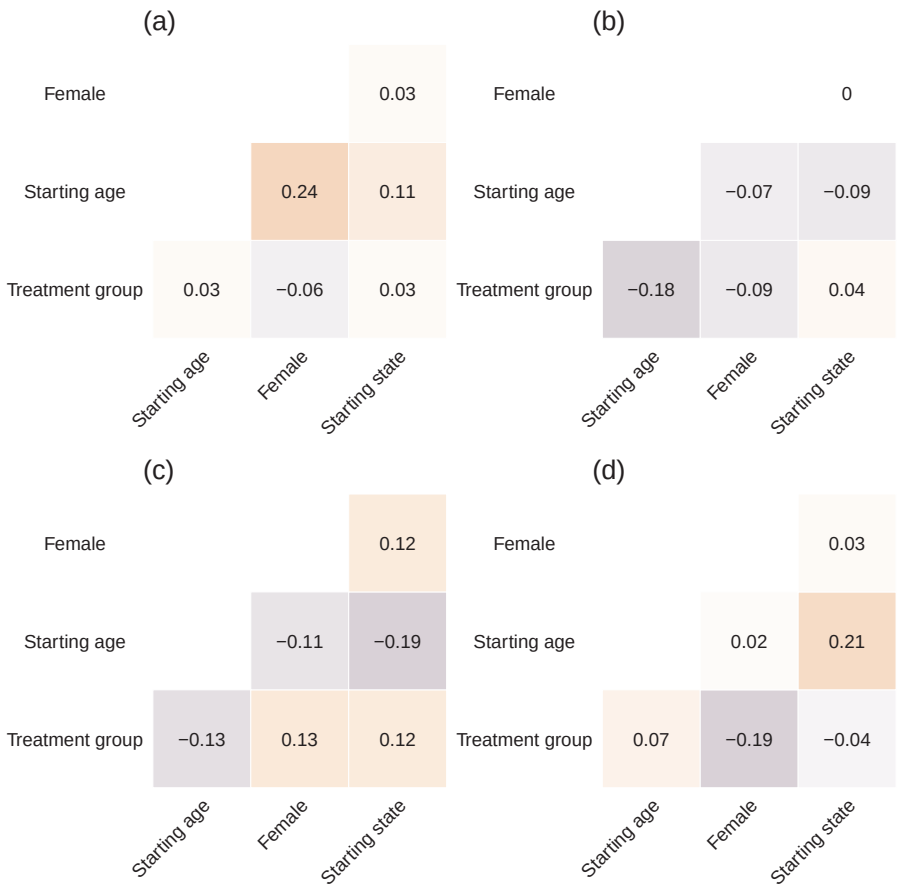


Figure 5.2: Spearman correlation among the variables $(\boldsymbol{x}, s_0)$ for the data sets $\mathcal{D}_r^{(train)}$ (a), $\mathcal{D}_s^{(0.5)}$ (b) $\mathcal{D}_s^{(0.5)}$ (c) and $\mathcal{D}_s^{(0.5)}$ (d).

maximum likelihood estimates from the training data $\mathcal{D}_r^{(train)}$. As we discussed in Section 2.5.3, such models are not differentially private, because the MLEs are not randomised. In Section 3.7, we discussed how a single sample from the posterior distribution of the parameters of a survival regression model can be differentially private when the likelihood is bounded. Further, we presented a novel method for differentially private WPHR models, and showed how this could be used for multi-state purposes if we treat each transition as independent. We use this method to generate differentially private synthetic processes, through estimating differentially private parameters from $\mathcal{D}_r^{(train)}$. As previously discussed, our method is limited to clock-reset MS-TTE models.

Recall that our method required that we divide the training data into subsets $\mathcal{D}_r^{\left(train_j\right)}$ representing each transition, and then we fit a posterior distribution for each transition j, using $\mathcal{D}_r^{\left(train_j\right)}$ as training data. We use *RStan* (Stan Development Team, 2023) to sample from posterior distributions $p\left(\boldsymbol{\theta_j}|\mathcal{D}_r^{\left(train_j\right)}\right)$ through MCMC algorithms. All the standard options in *RStan*

are followed. Thus, we have four chains and 2000 iterations, where the first 1000 are warm-up iterations.

Hyper-parameter tuning

We perform the hyper-parameter tuning by comparing the posterior distributions of the parameters of each transition to the gold standard, which is the MLEs of $\mathcal{D}_r^{(train)}$. We include a reduced version of Table 4.4 of these parameters in Table 5.3. Note that β_0 is transformed to the log scale for easier comparison.

Table 5.3: MLEs of the parameters of a **clock-reset** Weibull regression model fitted to the data set $\mathcal{D}_r^{(train)}$ for each of the $j \in 1, \ldots, 4$ transitions.

	j			
	1	2	3	4
λ	1.0417	1.0692	1.0516	0.6658
$\log(\beta_0)$	-4.9870	-6.5711	-0.8530	-5.0931
β_1	-0.1574	0.0900	0.3169	0.1138
β_2	0.0658	0.0713	0.0027	0.0662
β_3	0.1260	0.2764	0.1545	0.1614

First, we need to define the prior distributions of the parameters $(\lambda, \beta_0, \boldsymbol{\beta})$. We wish to use noninformative priors that adds uncertainty without influencing the location of the posterior distribution, and the choice must be made independent of the data. Nguyên and Hui (2018) use normal priors, and we do likewise. We use the following priors:

$$\beta_0, \beta_1, \beta_2, \beta_3, \log(\lambda) \sim N\left(0, 10^2\right).$$

Without considering the data, it is reasonable to assume that the effect of each parameter is centred around 0. The same holds for $\log(\lambda)$, because if $\log(\lambda) = 0$, then the hazard rate is constant.

The hyper-parameter η regulates how much each patient can contribute to the likelihood of each transition. As we discussed in Section 3.7, if the bound is too low, then only a few patients will be able to contribute with an uncut likelihood. As a first attempt, we follow the recommendation of Nguyên and Hui (2018) and set $\eta = \log(n)$, where $n = 391$ is the total number of patients in the training set. Through experiments on the liver cirrhosis data set, we find that this causes the RStan simulation to diverge, even for high values of ε. Still, we want η to be dependent upon the number of patients and not the total number of transitions, because we wish to control the contribution of each patient. From Equation (3.21), we see that the sanitiser function is fed an average likelihood across all the transition rows of a single patient. This secures that patients with a low number of total transitions can contribute more per transition than patients with many transitions. Consider Table A.1. The patient with *patient id* $= 2$ has more rows than the patient with *patient id* $= 1$, which means that we need each row of patient 1 to have a higher impact on the joint likelihood of that transition.

Note that we limit each patient's contribution to the likelihood separately for each transition. Then patients that contribute to each transition will have a

higher impact on the combined model than patients that only contribute to the likelihoods of some transitions. Consider again patient 1 and patient 2 from Table A.1. Patient 1 only contributes to two transition models, while patient 1 contributes to all.

Further experimenting with values of η leads to that $\eta = 3\log(n)$ results in convergence of all the transition models. We display the means and standard deviations of the posterior distributions of each parameter and for each transition $j \in \{1, \ldots, 4\}$ in Table 5.4. Each transition has $\varepsilon = 5/4$, resulting in a total privacy budget of $\varepsilon = 5$. We see that many of the mean values are far from the gold standard in Table 5.3. Because the standard deviations are high, all the MLEs are within one standard deviation of the posterior distributions and they are of the same magnitude.

Attempts to increase η or reintroducing an offset does not close the distance between the gold standard and the means of the posterior distributions. From Bayesian theory, the mode of the posterior distribution typically converges to the MLE when n reaches infinity (Gelman, 2013). For our adjusted likelihood, this is not the case. If $\eta \propto \log(n)$, then the lower bound of the sanitiser function in Equation (3.18) reaches infinity, but the upper bound is unchanged. Further research is needed to explore how increasing sizes of n influences the posterior mean.

Table 5.4: Means and standard deviations of the posterior distribution of the parameters, using hyperparameters $\eta = 3\log(n)$ and $\varepsilon = 5$.

	j			
	1	2	3	4
mean (λ)	1.2118	1.3936	1.1087	0.9432
sd (λ)	0.4247	0.6678	0.3517	0.2838
mean (β_0)	-6.5800	-7.9414	-1.5436	-5.3894
sd (β_0)	2.7902	4.0960	2.4085	3.0158
mean (β_1)	-0.2064	-0.1146	0.3986	-0.0923
sd (β_1)	1.0111	1.5710	0.9973	1.1142
mean (β_2)	0.0801	0.0774	0.0040	0.0655
sd (β_2)	0.0422	0.0626	0.0411	0.0454
mean (β_3)	0.1030	0.2994	0.1477	0.2287
sd (β_3)	1.0214	1.5785	0.9812	1.1585

Following from Equation (3.18) and (3.21), increasing the size of ε should lower the variance of the posterior distributions, as the weight is shifted from the prior distribution to the likelihood. We attempt to double the ε in Table 5.5, so that each transition has a privacy budget of $\varepsilon = 10/4$. We see that the variance decreases, but only slightly. Most of the parameters move slightly towards the gold standard displayed in Table 5.3.

The most substantial drawback to the Bayesian sampling mechanism is that we do not sample the posterior mean. For ε differential privacy, we are only allowed to draw a single sample from each posterior distribution, and this introduces even more uncertainty. We draw a single sample which we will now use to generate differentially private synthetic data based on the differentially private patients generated in the previous section. We choose to sample from the posterior distributions fitted with $\varepsilon = 5/4$, because we wish to reduce the

Table 5.5: Means and standard deviations of the posterior distribution of the parameters, using hyperparameters $\eta = 3log\,(n)$ and $\varepsilon = 10$.

	j			
	1	2	3	4
mean (λ)	1.1518	1.3064	1.0528	0.9295
sd (λ)	0.3009	0.4603	0.2432	0.1885
mean (β_0)	-6.1825	-7.5600	-1.2869	-4.9449
sd (β_0)	1.9437	2.8275	1.6709	2.0504
mean (β_1)	-0.1524	-0.0313	0.4018	-0.0218
sd (β_1)	0.6508	0.9832	0.6654	0.7495
mean (β_2)	0.0784	0.0806	0.0037	0.0629
sd (β_2)	0.0294	0.0429	0.0283	0.0309
mean (β_3)	0.0503	0.2386	0.1621	0.1584
sd (β_3)	0.6947	1.0580	0.6683	0.7444

total privacy budget, and the improvements of doubling the ε value were small. The sampled 5−differentially private parameters are displayed in Table 5.6.

Table 5.6: Sampled parameters of a 5−differentially private **clock-reset** Weibull regression model for each of the $j \in \{1, \ldots, 4\}$ transitions.

	j			
	1	2	3	4
λ	1.39	1.19	0.86	0.81
$\log(\beta_0)$	-8.57	-5.47	1.32	-4.08
β_1	0.43	-1.29	-0.22	-0.62
β_2	0.12	0.08	-0.04	0.06
β_3	-0.32	-1.59	1.16	-0.03

5.2 Evaluation of Differentially Private MS-TTE data

We can now use the 5−differentially private parameters from Table 5.6 and the 5−differentially private synthetic patients from Section 5.1.1 to generate a 10−differentially private synthetic data set, which we refer to as $\mathcal{D}_s$.

5.2.1 Kaplan-Meier Curves

We evaluate the utility by comparing the Kaplan-Meier curves of $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$, as we did in Section 4.4.2. The results are displayed in Figure 5.3. Even when the tabular data has low utility and the parameters are distant from the gold standard, only the first transition in Figure 5.3 show a significant difference between the survival curves of $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$, which suggests good utility. However, we should keep in mind that the Kaplan-Meier curves do not display differences between patients with different covariate values.

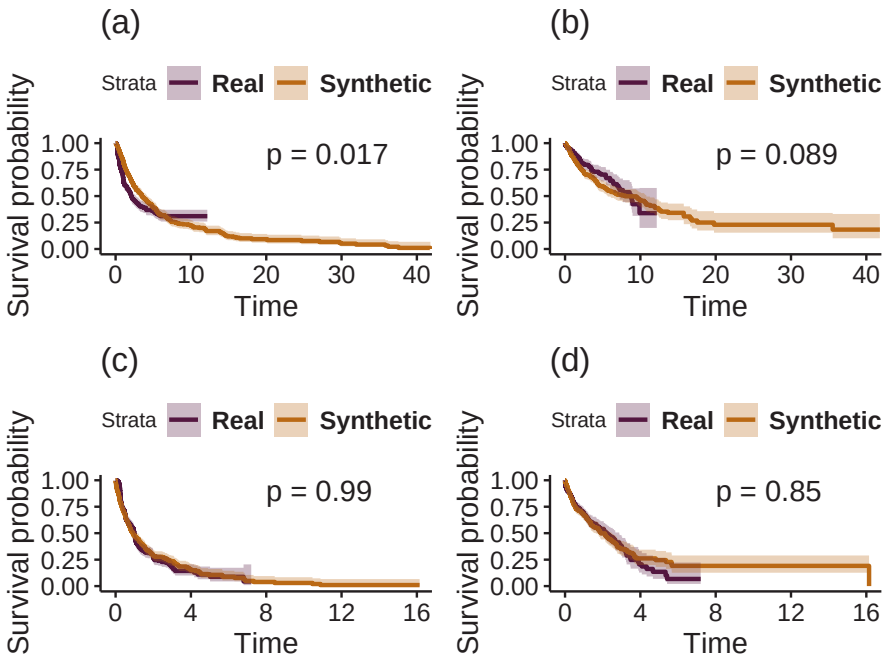


Figure 5.3: Kaplan-Meier survival curves with 95% confidence intervals and corresponding log-rank test p-values of the differentially private $\mathcal{D}_s$ and the training data $\mathcal{D}_r^{(train)}$ for each transition $k \in \{1, \ldots, 4\}$ in the figures *(a)-(d)*.

5.2.2 DCR and NNDR metrics

Finally, we use the DCR and NNDR distance metrics, which we introduced in Section 2.6, and demonstrated on non-differentially private data in Section 4.5.3. As before, we will now only consider uncensored data points. Our implementation makes it inconvenient to generate multiple differentially private synthetic data sets, compared to the approach in the previous chapter. Further details on this can be found in Appendix B. Therefore, we will only consider a single synthetic data set $\mathcal{D}_s$, which is equal in size to its training set $\mathcal{D}_r^{(train)}$. We compare the intra-data set distances of $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$, and the distance between $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$ in Table 5.7. When considering the $\alpha = 0.05$ metrics we see that the intra-data set distance of $\mathcal{D}_s$ is lower than for $\mathcal{D}_r^{(train)}$. This shows that the differentially private synthetic data contains some patients that are very similar, which indicates homogeneity. For $\alpha = 0.5$, $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$ are fairly equal, which shows that this is not an issue for $\mathcal{D}_s$ as a whole. Further, we see that the distances between $\mathcal{D}_s$ and $\mathcal{D}_r^{(train)}$ are greater than the intra-data set distances for both data sets. This holds for all distance metrics, and it indicates low utility and high privacy.

5.3 Discussion

Overall, we see that differential privacy causes a drop in utility compared to non-differentially private synthetic data, which is consistent with the theory

Table 5.7: DCR and NNDR distances with $w = 0.92$.

	DCR		NNDR	
α	0.05	0.5	0.05	0.5
$\|\mathcal{D}_s - \mathcal{D}_r^{(train)}\|$	0.0333	0.0714	0.4810	0.8284
$IDD\left(\mathcal{D}_s\right)$	0.0166	0.0508	0.3313	0.8112
$IDD\left(\mathcal{D}_r^{(train)}\right)$	0.0229	0.0544	0.4013	0.7792

on differential privacy that we covered in Section 2.5.3. The time-invariant variables from the tabular synthesiser had especially low utility, but this could be improved upon by using another differentially private synthesiser that is more suitable for tabular data. The MS-TTE synthesiser could also be improved upon by for example reducing the variance in the priors.

We have used a combined synthesiser with a privacy budget of $\varepsilon = 10$, which is considerably high. Recall Definition 2.5.1 and Equation (2.1) which defines $\varepsilon-$differential privacy. Say we have a mechanism $\mathcal{M}$ with $10-$differential privacy, an output S, and two data sets $\mathcal{D}$ and $\mathcal{D}'$ that differ by one row. Then the ratio between the probabilities that $\mathcal{M}$ outputs S is

$$\frac{P\left[\mathcal{M}\left(D\right) \in S\right]}{P\left[\mathcal{M}\left(D'\right) \in S\right]} \leq \exp\left(10\right) \approx 20000,$$

which indicates that the differing row can have a very high impact on the probability. At this point, the differentially private guarantee unfortunately holds little value, and we should attempt to lower the value of ε. Because there were only slight improvements of choosing larger ε values for the tabular synthesiser, it is possible that a lower ε choice is more sensible.

Ultimately, the choice of ε depends on the privacy-utility tradeoff, as the utility will decrease with smaller values of ε. Moreover, we need to consider the intended use. If the synthetic data should be used to fit Kaplan-Meier survival curves, then the performance of the differentially private synthetic data may be sufficient. However, if we need covariates with high utility because we wish to fit a Weibull or Cox regression model, then the non-differentially private synthesiser covered in Chapter 4 may be a better option.

CHAPTER 6

Concluding Remarks

In this thesis, we have explored the novel topic of synthetic multi-state time-to-event (MS-TTE) data. We have succeeded in proposing three different types of synthesisers tailored to MS-TTE data. First, we presented synthesisers based on both clock-forward and clock-reset multi-state models. Next, we extended the latter to a differentially private synthesiser, which offers a formal guarantee of the privacy of each individual data point.

Moreover, this thesis places synthetic MS-TTE data into the broader context of general synthetic data. We have surveyed a selection of privacy and utility evaluation methods which are tailored to tabular and survival data, and we have demonstrated how these methods can be extended to an MS-TTE setting. We argue that synthetic data evaluation should not only consider a specific synthetic data set. Instead, the evaluation process should also examine the privacy and utility properties of the synthesiser itself. To do so, we have proposed an evaluation framework that uses repeated synthesis to account for the variability in the synthesising process. This framework relies on prior work on the metrics DCR and NNDR (Zhao et al., 2021), which measure the distances between data sets based on single-point distances. In order to use these metrics, we required a method for measuring the distance between single MS-TTE data points. Therefore, we developed such a metric which can be applied to uncensored MS-TTE data.

We demonstrate the proposed synthesis models and the evaluation methods on a specific data set, the liver cirrhosis data, which is treated as our real data set. Our results indicate that the overall utility of the synthetic liver cirrhosis data is up to par. We observe that for some tasks, there is an added advantage of using the more complex clock-forward model as opposed to a clock-reset model. However, on most tasks, they perform equally well. Moreover, we have illustrated how the structure of the synthesiser can influence variable selection results. The privacy of the synthesisers was evaluated by measuring the distance between the synthetic data and their training sets. We have shown how this procedure revealed that the synthetic liver cirrhosis data is closer to its training data than unseen data, which indicates that the synthetic data could be at risk. Nonetheless, our staged membership inference attack (MIA) was unsuccessful. As we have only tested the synthesis and evaluation methods using a single data set, more research is needed before we can conclude that the synthesisers provide synthetic data with sufficient utility and privacy for MS-TTE data in general.

The differentially private MS-TTE synthesiser has covariates with low utility, and the parameters of the MS-TTE model have added noise. Nevertheless, the differentially private synthetic data is able to capture some of the transitions in a satisfactory manner. However, our results align with the literature (Stadler, Oprisanu and Troncoso, 2022) in that the differential privacy guarantee comes with an added penalty to the utility. In addition, we find that the ε value needs to be very high in order to achieve satisfactory results on most evaluation metrics. Still, the differentially private synthetic data may have sufficient utility for some specific use cases.

6.1 Further Work

This project could have taken many directions, as synthetic MS-TTE data has not been examined prior to this work. We discovered many tangents that we were unable to follow within the scope of this thesis, and several ideas for future research topics have emerged along the way. We conclude by outlining recommendations for further work.

Differential privacy

We developed a differentially private Weibull regression model as a stepping stone to a differentially private MS-TTE model. Further work is needed to evaluate how this model can be used for single-transition survival data and its asymptotic behaviour. In particular, it is advantageous to explore how η should best be defined and if the offset should be a function of the proportion of censored observations instead of η. Regarding the differentially private MS-TTE synthesiser, more work is needed on the effect of η and ε on privacy and utility and how the ε can be decreased without a significant loss in utility. In addition, it is necessary to further examine how increasing values of n impact the level of privacy and utility. Furthermore, we recommend experimenting with other differentially tabular synthesisers for generating the time-invariant variables, like PATE-GAN (Jordon, Yoon and M. v. d. Schaar, 2018).

More complex survival models

The choice of a Weibull regression model was made to ensure the overall simplicity of the synthesiser. Other models that do not assume a monotone hazard rate, such as the semi-parametric Cox regression and the non-parametric Kaplan-Meier, may provide a better fit. Another alternative is deep-learning approaches such as DeepHit (Lee et al., 2018), which is what SurvGAN (Norcliffe et al., 2023) is built on. DeepHit can handle time-dependent covariates, and instead of treating each transition separately and assuming independence, as we have done, it can create a joint competing risks model.

Moving beyond the Markov assumption

Even if we replace the Weibull regression models with DeepHit, the synthesiser will still rely on the Markov assumption, which can be oversimplified. We can add semi-time-dependent covariates such as time since the initial starting time and number of transitions. These covariates will remove the assumption, but this will still lose information about the complete state trajectories and transition times. We recommend that deep learning methods such as recurrent GANs, which have been used to generate synthetic time-series data (Esteban, Hyland and Rätsch, 2018), should be explored in the context of MS-TTE data.

Membership inference attacks

In Algorithm 3, we proposed a novel MIA attack, which uses hillclimbing in the domain space to search for observations in the training set. We also discuss how this attack can be used for attribute disclosure. This attack is computationally

demanding because we need to fit a new classifier for each point in the search. Therefore, we were not able to explore the potential impact of such an attack.

Testing on multiple data sets

Here, we have only used a single data set to test and evaluate the MS-TTE synthesis methods. To ensure that these methods and results hold for MS-TTE data sets in general, we recommend that a model comparison study with multiple data sets should be performed next.

Bibliography

Andersen, P. K. and Keiding, N. (1st Apr. 2002). 'Multi-state models for event history analysis'. In: *Statistical Methods in Medical Research* vol. 11, no. 2. Publisher: SAGE Publications Ltd STM, pp. 91–115.

Beaulieu-Jones, B. K. et al. (2019). 'Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing'. In: *Circulation Cardiovascular quality and outcomes* vol. 12, no. 7. Place: PHILADELPHIA Publisher: Lippincott Williams & Wilkins, e005122–e005122.

Behjati, R. et al. (May 2019). 'Synthetic Test Data Generation Using Recurrent Neural Networks: A Position Paper'. In: *2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*. 2019 IEEE/ACM 7th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), pp. 22–27.

Bellovin, S. M., Dutta, P. K. and Reitinger, N. (2019). 'Privacy and Synthetic Datasets'. In: vol. 22, p. 51.

Bianchi, F. M. et al. (2017). *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*. SpringerBriefs in Computer Science. Cham: Springer International Publishing.

Bonomi, L., Jiang, X. and Ohno-Machado, L. (1st Mar. 2020). 'Protecting patient privacy in survival analyses'. In: *Journal of the American Medical Informatics Association* vol. 27, no. 3, pp. 366–375.

Bowen, C. M. and Snoke, J. (3rd Feb. 2021). 'Comparative Study of Differentially Private Synthetic Data Algorithms from the NIST PSCR Differential Privacy Synthetic Data Challenge'. In: *Journal of Privacy and Confidentiality* vol. 11, no. 1. Number: 1.

Bozkaya, T., Yazdani, N. and Özsoyoğlu, M. (Jan. 1997). 'Matching and indexing sequences of different lengths'. In: *Proceedings of the sixth international conference on Information and knowledge management*. CIKM97: Sixth International Conference on Information and Knowledge Management. Las Vegas Nevada USA: ACM, pp. 128–135.

Breugel, B. van, Qian, Z. and Schaar, M. van der (8th July 2023). *Synthetic data, real errors: how (not) to publish and use synthetic data*. arXiv: 2305.09235 [cs].

Chen, D., Yu, N. et al. (2nd Nov. 2020). 'GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models'. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications*

Security. CCS '20. New York, NY, USA: Association for Computing Machinery, pp. 343–362.

Chen, D., Cheung, S.-c. S. et al. (Dec. 2021). 'Differentially Private Generative Adversarial Networks with Model Inversion'. In: 2021 IEEE International Workshop on Information Forensics and Security (WIFS). ISSN: 2157-4774, pp. 1–6.

Collett, D. (2015). *Modelling survival data in medical research*. Third edition. Chapman & Hall/CRC texts in statistical science series. Boca Raton: CRC Press.

Creswell, A. et al. (2018). 'Generative Adversarial Networks: An Overview'. In: *IEEE signal processing magazine* vol. 35, no. 1. Place: PISCATAWAY Publisher: IEEE, pp. 53–65.

Devore, J. L., Berk, K. N. and Carlton, M. A. (2021). *Modern Mathematical Statistics with Applications*. Springer Texts in Statistics. Cham: Springer International Publishing.

Deza, M. M. and Deza, E. (2014). *Encyclopedia of Distances*. Berlin, Heidelberg: Springer.

Drechsler, J. (2011). *Synthetic Datasets for Statistical Disclosure Control*. Vol. 201. Lecture Notes in Statistics. New York, NY: Springer New York.

Drechsler, J. and Reiter, J. P. (1st Dec. 2011). 'An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets'. In: *Computational Statistics & Data Analysis* vol. 55, no. 12, pp. 3232–3243.

Duncan, G. T., Elliot, M. and Salazar-González, J.-J. (2011). *Statistical Confidentiality: Principles and Practice*. New York, NY: Springer New York.

Dwork, C. and Lei, J. (31st May 2009). 'Differential privacy and robust statistics'. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. STOC '09. New York, NY, USA: Association for Computing Machinery, pp. 371–380.

Dwork, C. and Roth, A. (2014). *The Algorithmic Foundations of Differential Privacy*. Vol. 9. Foundations and Trends® in Theoretical Computer Science. ISSN: 1551-305X Issue: 3-4 Pages: 211–407. Norwell, MA: now.

Dwork, C. and Smith, A. (1st Apr. 2010). 'Differential Privacy for Statistics: What we Know and What we Want to Learn'. In: *Journal of Privacy and Confidentiality* vol. 1, no. 2. Number: 2.

El Emam, K. (2020). *Practical synthetic data generation: balancing privacy and the broad availability of data*. In collab. with Mosquera, L. and Hoptroff, R. 1st edition. Sebastopol, CA: O'Reilly Media.

Engelmann, J. and Lessmann, S. (15th July 2021). 'Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning'. In: *Expert Systems with Applications* vol. 174, p. 114582.

Esteban, C., Hyland, S. and Rätsch, G. (15th Feb. 2018). 'Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs'. In.

Fan, L. (2020). 'A Survey of Differentially Private Generative Adversarial Networks'. In.

Ferreira, J. A. and Zwinderman, A. H. (Aug. 2006). 'On the Benjamini–Hochberg method'. In: *The Annals of Statistics* vol. 34, no. 4. Publisher: Institute of Mathematical Statistics, pp. 1827–1849.

Figueira, A. and Vaz, B. (Jan. 2022). 'Survey on Synthetic Data Generation, Evaluation Methods and GANs'. In: *Mathematics* vol. 10, no. 15. Number: 15 Publisher: Multidisciplinary Digital Publishing Institute, p. 2733.

Gal, M. and Lynskey, O. (10th Apr. 2023). *Synthetic Data: Legal Implications of the Data-Generation Revolution*. Rochester, NY.

Gelman, A. (2013). *Bayesian Data Analysis, Third Edition, 3rd Edition*. In collab. with Carlin, J. et al. 3rd edition. CRC Press.

Goodfellow, I. J. et al. (10th June 2014). *Generative Adversarial Networks*. arXiv: 1406.2661 [cs, stat].

Gran, J. M. et al. (Dec. 2015). 'Causal inference in multi-state models–sickness absence and work for 1145 participants after work rehabilitation'. In: *BMC Public Health* vol. 15, no. 1, p. 1082.

Guillaudeux, M. et al. (10th Mar. 2023). 'Patient-centric synthetic data generation, no reason to risk re-identification in biomedical data analysis'. In: *npj Digital Medicine* vol. 6, no. 1. Number: 1 Publisher: Nature Publishing Group, pp. 1–10.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer.

Hernandez, M. et al. (July 2022). 'Synthetic data generation for tabular health records: A systematic review'. In: *Neurocomputing* vol. 493, pp. 28–45.

Hosmer, D. (2008). *Applied Survival Analysis: Regression Modeling of Time to Event Data, 2nd Edition*. In collab. with May, S. and Lemeshow, S. 1st edition. Wiley-Interscience.

Incerti, D. and Jansen, J. P. (8th Mar. 2021). *hesim: Health Economic Simulation Modeling and Decision Analysis*. arXiv: 2102.09437 [stat].

— (2nd Sept. 2022). *Markov and semi-Markov multi-state models*. URL: https://hesim-dev.github.io/hesim/articles/mstate.html (visited on 18/01/2023).

Jackson, C. (12th May 2016). 'flexsurv: A Platform for Parametric Survival Modeling in R'. In: *Journal of Statistical Software* vol. 70, pp. 1–33.

— (2022). 'Flexible parametric multi-state modelling with flexsurv'. In: *R package vignette* vol. 14, no. 4.

Jordon, J., Szpruch, L. et al. (1st May 2022). *Synthetic Data – what, why and how?* Publication Title: arXiv e-prints ADS Bibcode: 2022arXiv220503257J Type: article.

Jordon, J., Yoon, J. and Schaar, M. v. d. (27th Sept. 2018). 'PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees'. In: International Conference on Learning Representations.

Kalbfleisch, J. D. (2002). *The statistical analysis of failure time data*. In collab. with Prentice, R. L. 2nd ed. Wiley series in probability and statistics. Hoboken, N.J: Wiley-Interscience. XIII, 439.

Kauermann, G., Küchenhoff, H. and Heumann, C. (2021). *Statistical Foundations, Reasoning and Inference: For Science and Data Science*. Springer Series in Statistics. Cham: Springer International Publishing.

Kingma, D. P. and Welling, M. (1st May 2014). *Auto-Encoding Variational Bayes*. arXiv: 1312.6114 [cs, stat].

Kiran, A. and Kumar, S. S. (Mar. 2023). 'A Comparative Analysis of GAN and VAE based Synthetic Data Generators for High Dimensional, Imbalanced Tabular data'. In: *2023 2nd International Conference for Innovation in Technology (INOCON)*. 2023 2nd International Conference for Innovation in Technology (INOCON), pp. 1–6.

Kleinbaum, D. G. and Klein, M. (2005). *Survival Analysis: A Self-Learning Text*. 2nd ed. 2005. Statistics for Biology and Health. New York, NY: Springer New York : Imprint: Springer. 596 pp.

Kokosi, T. et al. (23rd May 2022). 'An overview on synthetic administrative data for research'. In: *International Journal of Population Data Science* vol. 7, no. 1. Number: 1.

Kotelnikov, A. et al. (30th Sept. 2022). *TabDDPM: Modelling Tabular Data with Diffusion Models*. arXiv: 2209.15421 [cs].

Kragh Andersen, P. et al. (1993). *Statistical Models Based on Counting Processes*. Springer Series in Statistics. New York: Springer. XI, 767.

Kundu, P., Darpe, A. K. and Kulkarni, M. S. (1st Dec. 2019). 'Weibull accelerated failure time regression model for remaining useful life prediction of bearing working under multiple operating conditions'. In: *Mechanical Systems and Signal Processing* vol. 134, p. 106302.

Kuppa, A., Aouad, L. and Le-Khac, N.-A. (2021). 'Towards Improving Privacy of Synthetic DataSets'. In: *Privacy Technologies and Policy*. Ed. by Gruschka, N. et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 106–119.

Laurence Goasduff (22nd June 2022). *Is Synthetic Data the Future of AI?* Gartner. URL: https://www.gartner.com/en/newsroom/press-releases/2022-06-22-is-synthetic-data-the-future-of-ai (visited on 10/10/2023).

Lee, C. et al. (26th Apr. 2018). 'DeepHit: A Deep Learning Approach to Survival Analysis With Competing Risks'. In: *Proceedings of the AAAI Conference on Artificial Intelligence* vol. 32, no. 1. Number: 1.

Lin, Z., Sekar, V. and Fanti, G. (18th Mar. 2021). 'On the Privacy Properties of GAN-generated Samples'. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics. ISSN: 2640-3498. PMLR, pp. 1522–1530.

Meira-Machado, L. et al. (Apr. 2009). 'Multi-state models for the analysis of time-to-event data'. In: *Statistical methods in medical research* vol. 18, no. 2, pp. 195–222.

Mirza, M. and Osindero, S. (6th Nov. 2014). *Conditional Generative Adversarial Nets*. arXiv: 1411.1784 [cs, stat].

Moore, A. W. (1990). *Efficient memory-based learning for robot control*. UCAM-CL-TR-209. University of Cambridge, Computer Laboratory.

NAV (1st Dec. 2019). *What is NAV?* nav.no. URL: https://www.nav.no/en/home/about-nav/what-is-nav (visited on 22/09/2022).

— (17th June 2022). *Test-Norge*. Testnav. URL: https://navikt.github.io/testnorge/applications/dolly/testnorge.html (visited on 07/10/2023).

— (26th May 2023). *Aktivitetsplan*. nav.no. URL: https://www.nav.no/aktivitetsplan (visited on 01/09/2023).

Nguyên, T. T. and Hui, S. C. (Aug. 2017). 'Privacy-Preserving Mechanisms for Parametric Survival Analysis with Weibull Distribution'. In: *2017 IEEE Trustcom/BigDataSE/ICESS*. 2017 IEEE Trustcom/BigDataSE/ICESS. ISSN: 2324-9013, pp. 456–463.

— (17th Oct. 2018). 'Privacy Protection for Flexible Parametric Survival Models'. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM '18. New York, NY, USA: Association for Computing Machinery, pp. 1273–1282.

Nikolenko, S. I. (2021). *Synthetic Data for Deep Learning*. Vol. 174. Springer Optimization and Its Applications. Cham: Springer International Publishing.

Norcliffe, A. et al. (11th Apr. 2023). 'SurvivalGAN: Generating Time-to-Event Data for Survival Analysis'. In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics. ISSN: 2640-3498. PMLR, pp. 10279–10304.

Norwegian Data Protection Authority (Jan. 2022). *NAV - sluttrapport*. Datatilsynet. URL: https://www.datatilsynet.no/regelverk-og-verktoy/sandkasse-for-kunstig-intelligens/ferdige-prosjekter-og-rapporter/nav-sluttrapport/ (visited on 15/09/2022).

Norwegian Ministry of Local Government and Modernisation (14th Jan. 2020). *National Strategy for Artificial Intelligence*. Regjeringen.no. URL: https://www.regjeringen.no/contentassets/1febbbb2c4fd4b7d92c67ddd353b6ae8/en-gb/pdfs/ki-strategi_en.pdf (visited on 16/09/2022).

Nowok, B., Raab, G. M. and Dibben, C. (2016). 'Synthpop: Bespoke creation of synthetic data in R'. In: *Journal of statistical software* vol. 74, no. 11. Place: LOS ANGELES Publisher: Journal Statistical Software, pp. 1–26.

Patki, N., Wedge, R. and Veeramachaneni, K. (Oct. 2016). 'The Synthetic Data Vault'. In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 399–410.

Ping, H., Stoyanovich, J. and Howe, B. (27th June 2017). 'DataSynthesizer: Privacy-Preserving Synthetic Datasets'. In: *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. SSDBM '17. New York, NY, USA: Association for Computing Machinery, pp. 1–5.

Qian, Z., Cebere, B.-C. and Schaar, M. van der (18th Jan. 2023). *Synthcity: facilitating innovative use cases of synthetic data in different data modalities*. arXiv: 2301.07573 [cs].

R Core Team (2023). *R: A language and environment for statistical computing*. manual. tex.organization: R Foundation for Statistical Computing. Vienna, Austria.

Rajabi, A. and Garibay, O. O. (2022). 'TabFairGAN: Fair Tabular Data Generation with Generative Adversarial Networks'. In: *Machine Learning and Knowledge Extraction* vol. 4, no. 2. Place: Basel Publisher: MDPI AG, pp. 488–501.

Rocher, L., Hendrickx, J. M. and Montjoye, Y.-A. de (2019). 'Estimating the success of re-identifications in incomplete datasets using generative models'. In: *Nature communications* vol. 10, no. 1. Place: LONDON Publisher: Springer Nature, pp. 3069–9.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. Wiley series in probability and mathematical statistics. Applied probability and statistics. New York: Wiley. XXIX, 258.

— (June 1993). 'Discussion Statistical Disclosure Limitation'. In: *Journal of Official Statistics* vol. 9, no. 2. Num Pages: 461 Place: Stockholm, Sweden Publisher: Statistics Sweden (SCB), p. 461.

Raab, G. M., Nowok, B. and Dibben, C. (2016). 'Practical Data Synthesis for Large Samples'. In: *Journal of Privacy and Confidentiality* vol. 7, no. 3. Number: 3, pp. 67–97.

— (13th Nov. 2021). *Assessing, visualizing and improving the utility of synthetic data*. arXiv: 2109.12717 [stat].

Sauber-Cole, R. and Khoshgoftaar, T. M. (22nd Aug. 2022). 'The use of generative adversarial networks to alleviate class imbalance in tabular data: a survey'. In: *Journal of Big Data* vol. 9, no. 1, p. 98.

Shlomo, N. (24th Dec. 2018). 'Statistical Disclosure Limitation: New Directions and Challenges'. In: *Journal of Privacy and Confidentiality* vol. 8, no. 1. Number: 1.

Shokri, R. et al. (May 2017). 'Membership Inference Attacks Against Machine Learning Models'. In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017 IEEE Symposium on Security and Privacy (SP). ISSN: 2375-1207, pp. 3–18.

Shumailov, I. et al. (31st May 2023). *The Curse of Recursion: Training on Generated Data Makes Models Forget*. version: 2. arXiv: 2305.17493 [cs].

Smith, A. (27th Sept. 2008). *Efficient, Differentially Private Point Estimators*. arXiv: 0809.4794 [cs].

Snoke, J. et al. (2018). 'General and specific utility measures for synthetic data'. In: *Journal of the Royal Statistical Society. Series A (Statistics in Society)* vol. 181, no. 3. Publisher: [Wiley, Royal Statistical Society], pp. 663–688.

Spinner, T. et al. (22nd Oct. 2018). 'Towards an Interpretable Latent Space – An Intuitive Comparison of Autoencoders with Variational Autoencoders'. In: Proceedings of the Workshop on Visualization for AI Explainability (VISxAI), p. 1.

Sramka, M. et al. (22nd Mar. 2010). 'A practice-oriented framework for measuring privacy and utility in data sanitization systems'. In: *Proceedings of the 2010 EDBT/ICDT Workshops*. EDBT '10. New York, NY, USA: Association for Computing Machinery, pp. 1–10.

Stadler, T., Oprisanu, B. and Troncoso, C. (24th Jan. 2022). *Synthetic Data – Anonymisation Groundhog Day*. arXiv: 2011.07018 [cs].

Stan Development Team (2023). *RStan: the R interface to Stan*. R package version 2.32.3.

Steinbakk, G. H., Langsrud, Ø. and Løland, A. (7th Jan. 2020). 'A brief overview of methods for synthetic data for official statistics'. In: *NR-notat (SAMBA/23/20)*, p. 22.

Sun, H. et al. (3rd June 2011). 'A New Similarity Measure Based on Adjusted Euclidean Distance for Memory-based Collaborative Filtering'. In: *Journal of Software* vol. 6, no. 6, pp. 993–1000.

Templ, M. (2017). 'Synthetic Data'. In: *Statistical Disclosure Control for Microdata: Methods and Applications in R*. Ed. by Templ, M. Cham: Springer International Publishing, pp. 157–179.

Therneau, T. M. (2023). *A Package for Survival Analysis in R*. R package version 3.5-7.

Tucker, A. et al. (2020). 'Generating high-fidelity synthetic patient data for assessing machine learning healthcare software'. In: *NPJ digital medicine* vol. 3, no. 1. Place: BERLIN Publisher: Springer Nature, pp. 147–147.

Vidnes Jensen, M. and Pihl Lyngstad, C. (8th Apr. 2019). *Innspill til strategi for kunstig intelligens*. URL: https://www.regjeringen.no/contentassets/0e36c85fcfe143a5b626c53cf292cb3b/strategi-kunstig-intelligens_innspill-fra-nav.pdf (visited on 15/09/2022).

Wang, Y.-X., Fienberg, S. and Smola, A. (1st June 2015). 'Privacy for Free: Posterior Sampling and Stochastic Gradient Monte Carlo'. In: *Proceedings*

of the 32nd International Conference on Machine Learning. International Conference on Machine Learning. ISSN: 1938-7228. PMLR, pp. 2493–2502.

Wreede, L. C. d., Fiocco, M. and Putter, H. (4th Jan. 2011). 'mstate: An R Package for the Analysis of Competing Risks and Multi-State Models'. In: *Journal of Statistical Software* vol. 38, pp. 1–30.

Xie, L. et al. (19th Feb. 2018). *Differentially Private Generative Adversarial Network*. arXiv: 1802.06739 [cs, stat].

Xu, C. et al. (Sept. 2019). 'GANobfuscator: Mitigating Information Leakage Under GAN via Differential Privacy'. In: *IEEE Transactions on Information Forensics and Security* vol. 14, no. 9. Conference Name: IEEE Transactions on Information Forensics and Security, pp. 2358–2371.

Xu, L., Skoularidou, M. et al. (27th Oct. 2019). *Modeling Tabular data using Conditional GAN*. arXiv: 1907.00503 [cs, stat].

Xu, L. and Veeramachaneni, K. (27th Nov. 2018). *Synthesizing Tabular Data using Generative Adversarial Networks*. arXiv: 1811.11264 [cs, stat].

Zhang, K., Patki, N. and Veeramachaneni, K. (28th July 2022). *Sequential Models in the Synthetic Data Vault*. arXiv: 2207.14406 [cs].

Zhang, Z. (Dec. 2016). 'Parametric regression model for survival data: Weibull regression model as an example'. In: *Annals of Translational Medicine* vol. 4, no. 24, p. 484.

Zhang, Z. et al. (Aug. 2018). 'Overview of model validation for survival regression model with competing risks using melanoma study data'. In: *Annals of Translational Medicine* vol. 6, no. 16. Number: 16 Publisher: AME Publishing Company, pp. 325–325.

Zhao, Z. et al. (31st May 2021). *CTAB-GAN: Effective Table Data Synthesizing*. arXiv: 2102.08369 [cs].

Aalen, O. O., Borgan, Ø. and Gjessing, H. K. (2008). *Survival and Event History Analysis*. Red. by Gail, M. et al. Statistics for Biology and Health. New York, NY: Springer.

Appendices

APPENDIX A

Liver Cirrhosis Data

Table A.1: Liver cirrhosis data of the first 5 patients in long format.

patient id	from	to	trans	T_{start}	T_{stop}	status	years	treatment group	starting age	female	starting state
1	2	1	3	0.00	0.41	0	0.41	0	73.92	0	2
1	2	3	4	0.00	0.41	1	0.41	0	73.92	0	2
2	2	1	3	0.00	0.69	1	0.69	0	47.93	1	2
2	2	3	4	0.00	0.69	0	0.69	0	47.93	1	2
2	1	2	1	0.69	1.19	1	0.50	0	47.93	1	2
2	1	3	2	0.69	1.19	0	0.50	0	47.93	1	2
2	2	1	3	1.19	2.00	1	0.81	0	47.93	1	2
2	2	3	4	1.19	2.00	0	0.81	0	47.93	1	2
2	1	2	1	2.00	4.75	1	2.76	0	47.93	1	2
2	1	3	2	2.00	4.75	0	2.76	0	47.93	1	2
2	2	1	3	4.75	5.72	1	0.97	0	47.93	1	2
2	2	3	4	4.75	5.72	0	0.97	0	47.93	1	2
2	1	2	1	5.72	6.75	0	1.04	0	47.93	1	2
2	1	3	2	5.72	6.75	1	1.04	0	47.93	1	2
3	2	1	3	0.00	0.85	1	0.85	0	41.72	0	2
3	2	3	4	0.00	0.85	0	0.85	0	41.72	0	2
3	1	2	1	0.85	6.02	1	5.16	0	41.72	0	2
3	1	3	2	0.85	6.02	0	5.16	0	41.72	0	2
3	2	1	3	6.02	7.07	1	1.06	0	41.72	0	2
3	2	3	4	6.02	7.07	0	1.06	0	41.72	0	2
3	1	2	1	7.07	13.39	0	6.32	0	41.72	0	2
3	1	3	2	7.07	13.39	0	6.32	0	41.72	0	2
4	2	1	3	0.00	0.79	0	0.79	0	76.84	0	2
4	2	3	4	0.00	0.79	1	0.79	0	76.84	0	2
5	2	1	3	0.00	0.75	0	0.75	0	79.89	1	2
5	2	3	4	0.00	0.75	1	0.75	0	79.89	1	2

Table A.2: Synthetic liver cirrhosis data generated with CTGAN.

patient id	from	to	trans	T_{start}	T_{stop}	status	years	treatment group	starting age	female	starting state
109	2	3	4	0.02	0.60	1	0.44	1	43.50	0	2
126	1	2	1	0.00	9.70	1	2.43	1	37.89	1	2
132	2	3	4	0.00	0.51	0	2.53	0	40.21	0	2
133	1	2	1	0.00	4.05	1	0.44	0	74.04	1	2
134	1	3	2	0.00	0.54	0	0.43	1	45.33	1	1
176	2	3	4	0.00	0.42	1	0.35	1	65.17	1	2
229	2	3	2	0.00	0.52	1	0.43	0	70.67	1	1
333	1	3	1	0.00	9.48	0	7.37	0	32.15	1	1
390	2	1	4	0.00	0.64	1	1.11	1	71.25	1	2
405	2	3	4	2.69	1.97	1	0.24	0	48.16	1	2
474	1	3	2	0.00	10.65	0	1.11	1	35.18	0	2
516	1	2	1	0.21	0.62	1	8.05	0	47.39	1	2
540	2	1	3	0.00	0.15	1	0.28	1	39.01	1	2
545	2	1	3	1.90	1.58	1	0.46	1	46.96	1	2
547	2	1	3	0.54	2.88	1	2.24	1	48.56	1	1
561	2	3	3	0.00	0.29	1	0.60	1	63.79	1	2
561	2	1	3	0.00	0.73	1	0.18	1	58.38	1	2
561	2	3	4	0.80	7.58	1	2.40	1	39.43	1	2
561	1	2	1	1.11	2.70	0	2.85	0	45.86	1	2
561	2	1	3	3.74	6.98	1	1.07	1	39.53	1	2

Table A.3: Synthetic liver cirrhosis data generated with CPAR.

patient id	from	to	trans	T_{start}	T_{stop}	status	years	treatment group	starting age	female	starting state
69131	2	2	3	0.55	1.18	0	1.53	1	78.46	1	2
69131	1	2	3	1.01	1.65	0	1.71	1	78.46	1	2
35127	1	2	2	1.15	1.22	0	1.68	1	74.14	1	2
35127	2	2	2	1.09	1.63	0	1.63	1	74.14	1	2
35127	1	2	3	1.25	2.03	0	1.64	1	74.14	1	2
35127	1	2	3	1.11	2.48	0	1.67	1	74.14	1	2
35127	1	2	2	0.98	2.90	0	1.66	1	74.14	1	2
85831	2	2	3	0.47	0.92	0	1.44	1	32.60	1	2
85831	1	2	3	0.13	1.60	0	1.65	1	32.60	1	2
85831	2	2	3	1.49	2.32	1	1.98	1	32.60	1	2
85831	2	2	3	2.21	2.61	1	1.36	1	32.60	1	2
85831	1	2	3	1.11	2.74	0	1.65	1	32.60	1	2
85831	1	2	3	1.83	2.81	0	1.65	1	32.60	1	2
30852	2	2	2	1.33	1.15	1	1.65	0	74.35	1	2
30852	1	2	3	0.82	1.70	0	1.58	0	74.35	1	2
98	1	2	3	0.84	1.24	0	1.53	1	78.43	1	1
98	2	2	3	1.15	1.65	0	1.71	1	78.43	1	1
98	2	2	3	1.22	2.13	0	1.65	1	78.43	1	1
98	2	2	3	1.17	2.55	0	1.59	1	78.43	1	1
98	1	2	2	0.97	3.05	0	1.66	1	78.43	1	1
98	2	2	3	1.21	3.46	0	1.68	1	78.43	1	1
98	2	2	3	0.93	3.89	0	1.65	1	78.43	1	1
98	2	2	3	1.12	4.26	0	1.60	1	78.43	1	1

Table A.4: Liver cirrhosis data of the first 7 uncensored patients in long format.

patient id	from	to	trans	T_{start}	T_{stop}	status	years	treatment group	starting age	female	starting state
1	2	1	3	0.00	0.41	0	0.41	0	73.92	0	2
1	2	3	4	0.00	0.41	1	0.41	0	73.92	0	2
2	2	1	3	0.00	0.69	1	0.69	0	47.93	1	2
2	2	3	4	0.00	0.69	0	0.69	0	47.93	1	2
2	1	2	1	0.69	1.19	1	0.50	0	47.93	1	2
2	1	3	2	0.69	1.19	0	0.50	0	47.93	1	2
2	2	1	3	1.19	2.00	1	0.81	0	47.93	1	2
2	2	3	4	1.19	2.00	0	0.81	0	47.93	1	2
2	1	2	1	2.00	4.75	1	2.75	0	47.93	1	2
2	1	3	2	2.00	4.75	0	2.75	0	47.93	1	2
2	2	1	3	4.75	5.72	1	0.97	0	47.93	1	2
2	2	3	4	4.75	5.72	0	0.97	0	47.93	1	2
2	1	2	1	5.72	6.75	0	1.04	0	47.93	1	2
2	1	3	2	5.72	6.75	1	1.04	0	47.93	1	2
4	2	1	3	0.00	0.79	0	0.79	0	76.84	0	2
4	2	3	4	0.00	0.79	1	0.79	0	76.84	0	2
5	2	1	3	0.00	0.75	0	0.75	0	79.89	1	2
5	2	3	4	0.00	0.75	1	0.75	0	79.89	1	2
6	2	1	3	0.00	0.77	0	0.77	0	60.68	1	2
6	2	3	4	0.00	0.77	1	0.77	0	60.68	1	2
7	2	1	3	0.00	0.55	0	0.55	1	78.23	1	2
7	2	3	4	0.00	0.55	1	0.55	1	78.23	1	2
10	2	1	3	0.00	0.15	0	0.15	1	75.31	1	2
10	2	3	4	0.00	0.15	1	0.15	1	75.31	1	2

APPENDIX B

Code

All source code for this thesis is available at:
https://github.com/ingriiser/Synthetic_MSTTE_Data

Chapter 2

The code used to generate the DCR and NNDR illustrations in Figure 2.1 and 2.5 are found in `notebooks/visualise_dcr_nndr.rmd`
In Example 2.6.2, we used the Iris data set to illustrate the pitfalls of removing exact synthetic matches. This experiment, along with plots of the corresponding figures, is provided in `notebooks/visualise_removing_synth.rmd`
Further, we provided an example on how distance measures can be used to expose privacy evaluation in Example 2.6.5. The calculations of the DCR and NNDR distances displayed in Table 2.1 and plots of the figures are found in `notebooks/demo_distance_expose_privacy_breach.rmd`

Chapter 3

All plots of the multi-state figures in Chapter 3 and 4 are made in `notebooks/state_trans_graphs.rmd`

The figure displaying the left-truncation of Weibull distributions in Figure 3.3 is plotted in `notebooks/trunc_weibull.rmd`

Chapter 4

The liver cirrhosis data is supplied in `data/liver_cirrhosis.csv` The wrapper functions that create different versions of this data set (like long and short form) are in `src/datasets_setup.r`
The first naive approach to generating synthetic liver cirrhosis data using CTGAN is provided in `src/naive_tabular.py` The next approach uses CPAR, and this is supplied in `src/naive_seq.py`
The code for generating synthetic MS-TTE clock-reset and clock-forward data is found in `src/generate_syn` The evaluation procedures are demonstrated over multiple notebooks. In `notebooks/distance_evaluation.rmd` there is a demonstration of the distance evaluation procedures, which uses evaluation functions in `src/distance_evaluation.r` Moreover, `notebooks/utility_evaluation.rmd` contains the utility evaluation and

uses functions found in `src/utility_evaluation.r` Lastly, the MIA experiment is in `notebooks/mi_attack.rmd` using functions in `src/mi_attack.r`

Chapter 5

The generation process of the differentially private synthetic data is described in `notebooks/generate_dp_syn.rmd` This notebook describes how the tabular data is generated in Python using DPGAN, which is done in `src/gen_patients.py` Then it demonstrates how the differentially private sampling from the posterior distribution is performed. The STAN script is in `src/dp_posterior_patient.stan` and the corresponding sampling in R is in `src/dp_posterior_sampling.r`
The evaluation of the differentially private data is performed in `notebooks/utility_dp.rmd` and `notebooks/evaluation_dp.rmd`