Title

Semi-Parametric Estimation in Network Data and Tools for Conducting Complex Simulation Studies in Causal Inference

Author

Sofrygin, Oleg

Publication Date

2016

Peer reviewed|Thesis/dissertation

Semi-Parametric Estimation in Network Data and Tools for Conducting Complex Simulation Studies in Causal Inference

by

Oleg A Sofrygin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Biostatistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Mark Van Der Laan, Chair
Professor Alan Hubbard
Assistant Professor John Marshall
Professor Jennifer Ahern

Spring 2016

Semi-Parametric Estimation in Network Data and Tools for Conducting Complex Simulation Studies in Causal Inference

Oleg A Sofrygin

Abstract

Semi-Parametric Estimation in Network Data and Tools for Conducting Complex Simulation Studies in Causal Inference

by

Oleg A Sofrygin

Doctor of Philosophy in Biostatistics

University of California, Berkeley

Professor Mark Van Der Laan, Chair

This dissertation is concerned with application of robust semi-parametric methods to problems of estimation in network-dependent data and the conduct of large-scale simulation studies for causal inference research in epidemiological and medical data. Specifically, Chapter 1 presents a modern semi-parametric approach to estimation of causal effects in a population connected by a single social network. The connectivity of the population units will typically imply that the observed data on these units is no longer independent and identically distributed. Moreover, such social settings typically result in highly dimensional data. This chapter contributes to current statistical methodology by presenting an approach that allows valid estimation and inference and addresses the statistical issues specific to such networked population datasets. The framework of semi-parametric estimation, called the targeted maximum likelihood estimation (TMLE), is presented. This framework improves upon the existing methods by offering robustness, weakened sensitivity to near positivity violations, as well as the ability to deal with high-dimensionality issues of social network data. In particular, this approach relies on the accurate reflection of the background knowledge available for a given scientific problem, allowing estimation and inference without having to make unrealistic assumptions about the structure of the data. In addition, this chapter generalizes previous work describing estimation of complex causal parameters, such as the direct treatment effects under interference and the causal effects of interventions on social network structure. Although the past decade has produced many contributions towards estimation of causal effects in social network settings, there has been considerably less research on the topic of variance estimation for such highly-dependent data. This chapter presents an approach to constructing valid inference, providing a variance estimator that is scalable to very large datasets with highly-connected observations. The efficient open-source software implementation of these methods also accompanies this chapter. Chapter 2 presents open-source software tools for conduct of reproducible simulation studies for complex parameters that emerge from application of causal inference methods in epidemiological and medical research. This simulation software is build on the framework of non-parametric structural

equation modeling. This chapter also studies simulation-based testing of statistical methods in causal inference for longitudinal data with time-varying exposure and confounding. It contributes to existing literature by presenting a unified syntax for non-parametrically defining complex causal parameters, which can be used as the model-free and agnostic gold standard for comparison of different statistical methods for causal inference. For instance, this chapter provides various examples of specification and evaluation of causal parameters that arise naturally in longitudinal causal effect analyses when using marginal structural models (MSMs). The application of these newly developed software tools to replication of several previously published simulation studies in causal inference are also described. Chapter 3 builds on the work described in Chapter 2 and addresses the issue of dependent data simulation for causal inference research in social network data. In particular, it provides a model-free approach to test the validity of various estimation procedures in simulated network-settings. This chapter first outlines a non-parametric causal model for units connected by a network and provides various applied examples of simulations with social network data. This chapter also showcases a possible application of the highly scalable open-source software implementation of the semi-parametric estimation methods described in Chapter 1. In particular, a large scale social network simulation study is described, and the performance of three dependent-data estimators from Chapter 1 is examined. This simulation study also examines the problem of inference for network-dependent data, specifically, by comparing the performance of the dependent-data TMLE variance estimator from Chapter 1 to the true TMLE variance derived from simulations. Finally, Chapter 3 concludes with a simulation study of an HIV epidemic described in terms of a longitudinal process which evolves over a static network in discrete time-steps among several highly inter-connected communities. The abstracts of the three works which make up this dissertation are reproduced below.

Chapter 1: This chapter describes the robust semi-parametric approach towards estimation and inference for the sample average treatment-specific mean in observational settings where data are collected on a single network of connected units (e.g., in the presence of interference or spillover). Despite recent advances, many of the currently used statistical methods rely on assumption of a specific parametric model for the outcome, even though some of the most important statistical assumptions required by these models are most likely violated in the observational network data settings, resulting in invalid and anti-conservative statistical inference. In this chapter, we rely on the recent methodological advances for the targeted maximum likelihood estimation (TMLE) for data collected on a single population of causally connected units, to describe an estimation approach that permits for more realistic classes of data-generative models and provides valid statistical inference in the context of such network-dependent data. The approach is applied to an observational setting with a single time point stochastic intervention. We start by assuming that the true observed data-generating distribution belongs to a large class of semi-parametric statistical models. We then impose some restrictions on the possible set of the data-generative distributions that may belong to our statistical model. For example, we assume that the dependence among units can be fully described by the known network, and that the dependence on

other units can be summarized via some known (but otherwise arbitrary) summary measures. We show that under our modeling assumptions, our estimand is equivalent to an estimand in a hypothetical IID data distribution, where the latter distribution is a function of the observed network data-generating distribution. With this key insight in mind, we show that the TMLE for our estimand in dependent network data can be described as a certain IID data TMLE algorithm, also resulting in a new simplified approach to conducting statistical inference. We demonstrate the validity of our approach in a network simulation study. We also extend prior work on dependent-data TMLE towards estimation of novel causal parameters, e.g., the unit-specific direct treatment effects under interference and the effects of interventions that modify the initial network structure.

Chapter 2: This chapter introduces the **simcausal** R package - an open-source software tool for specification and simulation of complex longitudinal data structures that are based on non-parametric structural equation models. The package aims to provide a flexible tool for simplifying the conduct of transparent and reproducible simulation studies, with a particular emphasis on the types of data and interventions frequently encountered in real-world causal inference problems, such as, observational data with time-dependent confounding, selection bias, and random monitoring processes. The package interface allows for concise expression of complex functional dependencies between a large number of nodes, where each node may represent a measurement at a specific time point. The package allows for specification and simulation of counterfactual data under various user-specified interventions (e.g., static, dynamic, deterministic, or stochastic). In particular, the interventions may represent exposures to treatment regimens, the occurrence or non-occurrence of right-censoring events, or of clinical monitoring events. Finally, the package enables the computation of a selected set of user-specified features of the distribution of the counterfactual data that represent common causal quantities of interest, such as, treatment-specific means, the average treatment effects and coefficients from working marginal structural models. The applicability of **simcausal** is demonstrated by replicating the results of two published simulation studies.

Chapter 3: The past decade has seen an increasing body of literature devoted to the estimation of causal effects in network-dependent data. However, the validity of many classical statistical methods in such data is often questioned. There is an emerging need for objective and practical ways to assess which causal methodologies might be applicable and valid in such novel network-based datasets. In this chapter we describe a set of tools implemented as part of the **simcausal** R package that allow simulating data based on the non-parametric structural equation model for connected units. We also provide examples of how these simulations may be applied to evaluation of different statistical methods for estimation of causal effects in such data. In particular, these simulation tools are targeted to the types of data and interventions frequently encountered in real-world causal inference research in social networks, such as, observational studies with spill-over or interference. We developed a novel R language interface which simplifies the specification of network-based functional relationships between connected units. Moreover, this network-based syntax can be combined with the

syntax for specifying longitudinal data structures, allowing for simulations of network-based processes that evolve in time (e.g., contagion in epidemic modeling). We provide various examples of simulation studies that involve units connected by various network models. These simulations were designed to mimic the types of studies one might conduct in real life with the aim of answering specific causal public health questions. We also demonstrate one application of these new tools by conducting a simulation study that compares the performance of three estimators of the counterfactual mean outcome in a network-dependent data setting. Finally, we describe a simulation study with longitudinal data that mimics a spread of HIV epidemic over time for highly inter-connected communities.

To my partner and best friend, Monika, without whom nil.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank my advisor, Mark van der Laan, for his extraordinary support, tireless encouragement and guidance during my time at Berkeley. Mark's enthusiasm for science captivated me from the first day I sat in on his class. His work ethic and passion for science have been an example and an inspiration to me. I am grateful to Mark for being always available and eager to help me. I am very thankful for all the life lessons that Mark has taught me, which extend far beyond statistics and academia. I am also grateful for his trust in me and my work and for allowing me the complete intellectual freedom to find and pursue my scientific interests. I consider myself incredibly lucky to be one of Mark's students.

I thank my mentor, Romain Neugebauer, to whom I owe heartfelt thank you for investing so much time in my work, for incredibly useful and fun conversations, for his guidance, his patience and for spending so much time editing my writing and providing suggestions on how to improve it.

I thank my other mentors, Alan Hubbard and John Marshall, for their encouragement and so many contributions to my education. A very special thanks to Sharon Norris, for her vital work in running the department and saving students like myself from endless bureaucratic hurdles and missed deadlines.

I thank my incredible fellow students and friends that surrounded me throughout my education, including Erin Ledell, Samuel Lendle, Alex Luedtke, Robin Mejia, Jeremy Coyle, Lucia Petito, Marla Johnson, Molly Davies, Wenjing Zheng, Luca Pozzi, Sara Moore, Jonathan Levy, Jonathan Pollack, Mary Combs and Ivan Diaz. Thank you for all the laughs and all the learning we did together. I am also thankful to my new friends outside of academia who provided me with much needed emotional support during the last three years, especially David and Dave, Kyle Work, Ryan and Eddie, and Jonathan Liew.

I thank my collaborator, Elizabeth (Betsy) Ogburn, for her encouragement, great spirit and so many helpful discussions.

I am especially thankful to my parents, Andrei Vasil'evich Sofrygin and Tatiana Viktorovna Sofrygina, for their love and support. I am eternally grateful to you both.

Finally I thank my wonderful partner, my wife and my best friend, Monika. Thank you for your endless support and your continued belief in me. Thank you for following me across the continent. Thank you for always being there. It is thanks to you and only you that I have the freedom to pursue my life dreams. It is also thanks to you and only you that I stand a chance of succeeding in this pursuit.

Chapter 1

Semi-Parametric Estimation in Network Data

1.1 Introduction

Motivation

In this chapter we are concerned with estimation and inference for the sample average treatment effect [80] in an observational setting that involves members of a single connected network. Valid statistical inference in such settings presents a number of significant challenges. For example, the frequently made assumption of independence among units is generally violated when data is collected on a population of connected units, since the network interactions will often cause the exposure of one unit to have an effect on the outcomes of other connected units. In general, statistical methods for estimation and inference in observational network data are faced with three key challenges that set such data apart from the classical statistical methods for independent observational data: (i) the outcome for each unit can be a function of the treatment assignments of other units that are connected to the unit through its network, an occurrence referred to as interference or spillover [57, 113]; (ii) the outcome of each unit can be a function of the baseline covariates of other units that are connected to the unit through its network, sometimes referred to as network-correlated outcomes [9]; and (iii) the observed exposure allocation for each unit can be a function of the baseline covariates of other units. As a result, the sample units are not independent, and, in fact, one only observes a single draw from the true data generating distribution. Therefore, classical statistical methods that assume independence among the observed outcomes will be often overly optimistic and invalid for quantifying the variability of estimators in such data. In addition, many of the current estimation procedures for observational network data assume a particular class of parametric or restrictive classes of semi-parametric models for the observed data-generating distribution, which makes these methods highly susceptible to bias due to model misspecification [26, 27].

Targeted maximum likelihood (or minimum loss-based) estimation (TMLE) [67, 66] is a

general framework for constructing asymptotically linear and efficient substitution estimators, that belong to a much larger class of semi-parametric models, while providing asymptotically valid statistical inference. Recently, the TMLE framework has been extended to estimation of treatment effects in dependent observational data [63], where the dependence among units is described by the network of connections formed by these units (e.g., social or geographical networks). Our aim will be to provide an accurate reflection of the background knowledge available for a given scientific problem, while still being able to perform valid statistical estimation. Thus, we start by assuming a realistic semi-parametric statistical model for the generating distribution of observed network data, which places minimal restrictions on the set of such possible data-generating distributions. The first objective of this chapter is to apply the TMLE framework to estimation of causal effects in single time-point observational network data. Our next objective is to verify the practical validity of our approach with a simulation study. We demonstrate that consistent estimation and valid asymptotic inference of the sample average treatment effects for a single time point stochastic interventions is possible in this larger class of semi-parametric models, even in observational network data where the dependence between units is induced by the known network structure.

Brief review of relevant literature

The literature on networks and causal inference in network data is rapidly evolving. However, the existing statistical methods for performing estimation and inference for causal effects in networks are limited and the literature on this subject has only recently started to develop [63, 130, 83, 128, 120]. Our review is not intended to be exhaustive, instead, we focus on the key aspects and challenges of statistical estimation of treatment effects in observational network data. Most of the recently proposed approaches can be categorized as relying on either the assumption of randomized exposures across units [107, 4, 18, 133, 2, 3, 121, 69, 25, 9], or on parametric modeling of the outcome as a particular function of the unit's network. Some of the parametric approaches applied in the network settings include generalized linear models (GLMs) and generalized estimating equations (GEEs) [26, 27], methods which have important limitations [70, 127, 129, 125, 83]. For one, GLMs and similar modeling techniques require making strong, simplifying modeling assumptions about the underlying data generating process. Hence, model misspecification for GEEs and GLMs in the network data settings is a major cause of concern. Perhaps more importantly, performing valid statistical inference with GLMs and other similar statistical techniques generally requires independence of the observational units, an assumption that is unlikely to hold due to the very nature of the network data. It has also been previously described that application of such standard statistical procedures to dependent data will result in invalid and generally anti-conservative statistical inference [70, 83].

In addition, a few promising methodological approaches to estimation in network data have begun to emerge in recent years. For example, [4] proposed a Horvitz-Thompson estimator in a randomized study settings, defined the so-called "network exposure model" and derived the finite sample estimator of the variance. However, such methods are of limited

utility in observational settings. Other proposed approaches for identification and estimation of treatment effects in networks include stochastic actor-oriented models [119], and a linear Bayesian modeling approach that can accommodate for network uncertainty [121]. Another recently proposed approach applied the semi-parametric framework of targeted maximum likelihood estimation to the observation network data settings [63], yielding valid asymptotic inference, while allowing for a much larger and realistic class of data-generative models. We apply the latter approach in the sections that follow.

Contributions and organization of this chapter

We start by describing the type of data that may arise in an observational study on a population of connected units. Consider a study in which we observe a sample of N dependent units. For each unit we collect baseline covariates, a binary exposure, and a one-dimensional outcome of interest. We denote the sample by the random vector $\mathbf{O} = (\mathbf{W}, \mathbf{A}, \mathbf{Y}) \sim P_0$, where $\mathbf{W} = (W_i)_{i=1}^N$ is a vector of baseline covariates across all units, $\mathbf{A} = (A_i)_{i=1}^N$ is a vector of exposures, $\mathbf{Y} = (Y_i)_{i=1}^N$ is a vector of outcomes, and P_0 belongs to a large semi-parametric model. We assume each W_i has finite support, each A_i is binary, and Y_i is either binary (e.g., indicating survival beyond a specific time point, or the success of a particular intervention) or bounded (e.g., a count of the number of times an event of interest has occurred during the follow-up period, or a continuous measure of a biomarker level at the end of the study). For each unit i in the sample, we also collect the information on other units in $\{1, \ldots, N\} \backslash \{i\}$ that are connected to (or influence) i. These units are referred to as "*i's friends*", and this set is denoted by $F_i \subseteq \{1, \ldots, N\}$. It is assumed that F_i is recorded at baseline, along with other baseline covariates, and it is assumed fixed. Additionally, we allow $|F_i|$, the number of friends for unit i, to vary in i, but assume that this number is bounded by some known global constant K that doesn't depend on N. The vector $\mathbf{F} = (F_i)_{i=1}^N$ is then referred to as the "*network profile*" of $\mathbf{O}$. For example, in an experiment evaluating the effects of introducing a new service to an online social network, for each unit, F_i could denote the set of all online friends of i, whose exposure status may influence i's outcome. Alternatively, in a study of the effects of early HIV treatment initiation, F_i could be the set of all sexual partners of unit i. We allow for the following types of between-unit dependencies: (i) the unit-level exposures can depend on baseline data of itself and other units, and (ii) unit-level outcomes can depend on baseline and exposure data of itself and other units. An important ingredient of our modeling approach is to assume that the dependence of each unit i on other units is fully described by the network. Specifically, we assume that the dependence of i's treatment and outcome on other units is limited to the set of i's friends. A second important ingredient is the assumption that these dependencies can be accurately described with some known summary measures, which map the data collected on friends of each unit into a summary that has the same dimension for all units.

We now wish to estimate and perform valid inference for the sample-average of the unit-specific mean outcomes, given as $1/N \sum_{i=1}^N E\left[E_{\mathbf{g}^*}\left(E(Y_i \mid \mathbf{A}, \mathbf{W}) \mid \mathbf{W}\right)\right]$, where the N exposures are assigned according to some user-specified stochastic intervention $\mathbf{g}^*$. That is, $\mathbf{g}^*$ is a

fixed conditional density for drawing exposures $\mathbf{A}$, given baseline covariates $\mathbf{W}$. Under additional causal assumptions, this statistical quantity will be equal to the sample average of the expected counterfactual outcomes, defined as the expected outcomes which would have been obtained if the units in the sample had actually been treated according to the treatment regime specified by $\mathbf{g}^*$ [63]. We note that the definition of intervention $\mathbf{g}^*$ is kept general, allowing for any static, dynamic, or stochastic single time point interventions. We also note that the definition of the statistical parameter can be extended to multiplicative or additive sample average treatment effects [80, 6]. Additionally, our statistical parameter is defined with respect to a given network profile $\mathbf{F}$ and given sample size N, and thus should be regarded as a type of a data-adaptive statistical parameter [65], since its true value is allowed to change for different sample sizes and different network structures. Finally, we note that the statistical parameter defined in this manner has a generally meaningful statistical interpretation, even when the required causal assumptions do not hold, and our focus is only on the aspects of statistical estimation of such parameters in the context of the semi-parametric modeling framework.

The main contributions of this chapter are as follows: We start by pointing out that our statistical parameter can be represented in a novel way, in light of the fact that it depends on the joint distribution of the observed data only via a mixture of the unit-specific distributions, and we will use $\bar{P}$ to denote this mixture distribution. In particular, we show that our dependent-data parameter can be represented as a mapping $\bar{\Psi}$ from mixture $\bar{P}$, giving our parameter an alternative interpretation as a G-computation formula for the mean of the iid outcomes generated from the post-intervention distribution under some fixed stochastic intervention $\bar{g}^*$, i.e., $\bar{\Psi}(\bar{P}_0) = E\bar{Y}_{\bar{g}^*}$. This mixture representation then also leads us to conclude that the estimation of $\bar{\Psi}(\bar{P}_0)$ should only be concerned with estimation of the relevant factors of the mixture $\bar{P}$ and we use this fact to provide a self-contained description of the semi-parametric estimation framework in network-dependent data developed by [63]. As we will show, this new mixture representation implies that our statistical parameter can be estimated by simply ignoring dependence among units and treating them as if they are independent and identically distributed (iid), suggesting that a large class of iid-data estimators is applicable to estimation problems such as the one we describe in this chapter. Based on this key insight, the dependent-data TMLE from [63] is then presented as a typical iid-data TMLE. We also apply the new mapping $\bar{\Psi}$ for performing statistical inference, presenting a new robust asymptotic variance estimator which improves upon the previously proposed estimator from [63] in that it remains conservative even under the outcome model misspecification and no longer requires the assumption of complete independence among baseline covariates. We then conduct a simulation study to provide a proof of concept for our framework and to assess the feasibility of unbiased estimation and inference in finite sample observational network data. We also compare the performance of TMLE to other statistical procedures using a newly developed **R** package *tmlenet* [116]. Finally, this chapter generalizes the previously described dependent-data TMLE framework to allow estimation of novel causal parameters. In particular, we describe the TMLE for estimating the treatment effect under arbitrary unit-specific stochastic interventions on N groups of friends, which

may be incompatible with the existence of a single multivariate stochastic intervention on all N units of study, e.g., interventions that characterize the direct treatment effect under interference. We also extend our framework to allow estimation of parameters defined by interventions that statically or stochastically modify the initial network structure.

The rest of the chapter is organized as follows: We start by formally describing the observed network data, defining the statistical model, and defining the statistical parameter of interest in Section 1.2. Next, in Section 1.3, we describe an alternative representation of our statistical parameter as a mapping from some mixture distribution, derived as a function of the actual observed data distribution. In Section 1.4, we describe the iid TMLE algorithm for estimating the dependent-data sample-average treatment effects under single time-point stochastic intervention $\mathbf{g}^*$. We then proceed by proposing a new estimator of the asymptotic variance for this TMLE in Section 1.5. Next, we describe a simulation study that examines the finite sample performance of the proposed TMLE, and that of the new variance estimator, in Section 1.6. We then describe in Section 1.7 how our framework generalizes to estimation of parameters indexed by arbitrary collections of stochastic interventions or by interventions on the network structure. We conclude with a discussion of the relative merits and limitations of our proposed approach in Section 1.8.

A note on notation

Throughout this chapter we use the bold font capital letters, such as $\mathbf{O}$, to denote random vectors that include observations on all N units, and bold font small letters, such as $\mathbf{o}$, to denote their corresponding fixed values. For example, $\mathbf{A}$ will denote the vector of N exposures, i.e., $\mathbf{A} = (A_1, \ldots, A_N)$. We will also use the standard font capital letters with a subscript to denote the unit-specific observations, i.e., A_i will denote the exposure for the unit i. Finally, we will use the over-bar symbol to denote mixture distributions across all N units, as well as their corresponding random variables. For example, $\bar{P}_W$ will denote the mixture of N unit-specific distributions of baseline covariates W_i, for $i = 1, \ldots, N$, i.e., $\bar{P}_W = 1/N \sum_{i=1}^{N} P_{i,0}$, and $\bar{W}$ will denote a random variable distributed according to $\bar{P}_W$. The only exception to this rule will be $\bar{Q}$, which will denote the conditional expectation of the unit-specific outcome Y_i, as well as the conditional expectation of the mixture-based outcome $\bar{Y}$, which happen to be equal under our statistical model.

1.2 Statistical model and parameter

Suppose P_0^N is the true data generating distribution for N observed and connected units, with $\mathbf{O} = (\mathbf{W}, \mathbf{A}, \mathbf{Y}) \sim P_0^N$ denoting the random vector for these N units and $O_i = (W_i, A_i, Y_i)$, for $i = 1, \ldots, N$. The network profile $\mathbf{F}$ is assumed recorded at baseline, i.e., $\mathbf{F} \in \mathbf{W}$. We also assume all Y_i are bounded random variables. Let $\mathcal{M}$ denote a statistical model containing P_0^N. Since $\mathbf{O}$ represents a network of dependent units, we observe only a single draw from P_0^N, and as a result, are unable to estimate P_0^N from this single observation $\mathbf{O}$. We now

proceed by making a series of statistical assumptions, which will allow us to learn the true distribution of $\mathbf{O}$ based on this single draw. In particular, we introduce these assumptions by making restrictions on the set of possible distributions that belong $\mathcal{M}$. We will then define our statistical quantity of interest as a mapping Ψ from $\mathcal{M}$ into the real line $\mathbb{R}$.

Following [63], we make the following set of statistical assumptions for any $P_0^N \in \mathcal{M}$:

A1. Conditional on $\mathbf{F}$, each W_i depends on at most K other observations in $\mathbf{W} = (W_1, \ldots, W_N)$, i.e., if $(W_j : j \in S_i)$ is the set of all observations dependent with W_i then $\max_i |S_i| \leq K$ and K must not depend on N;

A2. $\mathbf{A} = (A_1, \ldots, A_N)$ are independent, conditional on $\mathbf{W}$;

A3. $Y_1, \ldots, Y_N$ are independent, conditional on $(\mathbf{A}, \mathbf{W})$.

These assumptions imply the following likelihood for $P^N \in \mathcal{M}$:

$$p^N(\mathbf{O}) = \left[\prod_{i=1}^{N} p_{Y_i|\mathbf{A},\mathbf{W}}(Y_i \,|\, \mathbf{A}, \mathbf{W})\right] \left[\prod_{i=1}^{N} p_{A_i|\mathbf{W}}(A_i \,|\, \mathbf{W})\right] p_{\mathbf{W}}(\mathbf{W}).$$

We also assume conditional independence implied from the known network structure:

A4. Assume that conditional distributions $P(Y_i \,|\, \cdot)$ only depend on $(A_j, W_j : j \in F_i^*)$, for $F_i^* = F_i \cup \{i\}$, and similarly, $P(A_i \,|\, \cdot)$ depend on $(W_j : j \in F_i^*)$.

We now introduce the dimension reducing assumptions for these conditional distributions. Specifically:

B1. Assume that each $P(A_i \,|\, \cdot)$ is a function of some fixed-dimension summary measure $w_i^s((W_j : j \in F_i^*))$, and each $P(Y_i \,|\, \cdot)$ is a function of fixed-dimension summary measures $a_i^s((A_j, W_j) : j \in F_i^*)$ and $w_i^s((W_j : j \in F_i^*))$. We assume $w_i^s(\cdot)$ and $a_i^s(\cdot)$ are known functions that map into Euclidean set of constant (in i) dimension that does not depend on N, where a_i^s map into some common space $\mathcal{A}^s$, and w_i^s map into some common space $\mathcal{W}^s$.

Formally, these summary measures for $i = 1, \ldots, N$ are defined as:

$$\begin{aligned} W_i^s &= w_i^s(\mathbf{W}) = w_i^s(W_j : j \in F_i^*) \in \mathcal{W}^s, \\ A_i^s &= a_i^s(\mathbf{A}, \mathbf{W}) = a_i^s((A_j, W_j) : j \in F_i^*) \in \mathcal{A}^s, \end{aligned}$$

where above we also introduced the shorthand notation W_i^s and A_i^s, for $w_i^s(\mathbf{W})$ and $a_i^s(\mathbf{A}, \mathbf{W})$, respectively. As an example of such summary measures, an investigator conducting a social networks study might be willing to assume that the outcomes Y_i depend on $(\mathbf{A}, \mathbf{W})$ only through summary measures $(a_i^s(\mathbf{A}), w_i^s(\mathbf{W}))$, where $a_i^s(A) = (A_i, A_i^c)$ and $w_i^s(\mathbf{W}) = (W_i, W_i^c)$ and A_i^c is some one dimensional summary of exposures of i's friends and W_i^c is some one dimensional summary of baseline covariates of i's friends, where A_i^c is the same function

in i and W_i^c is also the same function in i. If one is unwilling to make such strong dimensionality reducing assumptions, one could instead assume $a_i^s(\mathbf{A}) = (A_j : j \in F_i^*)$ and $w_i^s = (W_j : j \in F_i^*)$, without assuming a particular functional form of a_i^s and w_i^s. By filling the empty spots in $a_i^s(\cdot)$ and $w_i^s(\cdot)$ with missing values one would assure that all summaries $(a_i^s(\cdot), w_i^s(\cdot))$ are of constant dimension across i and that the information on the number of friends of i is also captured. In summary, we allow A_i^s and W_i^s to be arbitrary functions of the units' network, as long as their dimension is fixed, common-in-i, and doesn't depend on N. Applying these summary measures to the observed data, we obtain the following likelihood:

$$p^N(\mathbf{O}) = \left[\prod_{i=1}^N p(Y_i \,|\, A_i^s, W_i^s)\right] \left[\prod_{i=1}^N p(A_i \,|\, W_i^s)\right] p(\mathbf{W}).$$

We are now ready to make the final set of restrictions on $\mathcal{M}$. Specifically:

C1. Assume that all Y_i are sampled from the same distribution Q_Y with density given by $q_Y(Y_i \,|\, a^s, w^s)$, conditional on fixed values of the summary measures (A_i^s, W_i^s), for $i = 1, \ldots, N$. Similarly, assume that all A_i are sampled from the same distribution given by density $g(A_i \,|\, w^s)$, conditional on some fixed value of the summary measures $W_i^s = w^s$, for $i = 1, \ldots, N$.

We also assume (without loss of generality) that the densities g and q_Y are well-defined with respect to some dominating measure. Using the previous example of the summary measures, i.e., $W_i^s = (W_i, W_j : j \in F_i)$ and $A_i^s = (A_i, A_j : j \in F_i)$, this assumption implies that the units i and j will be subject to the same conditional distributions for drawing their treatment and outcome, if i and j have the same number of friends, same individual covariate and treatment values, and the same values for the covariates and treatments of their friends. This implies that its possible to learn the common-in-i densities Q_Y and g from a single (but growing) draw $\mathbf{O}$ from P_0^N as $N \to \infty$, resulting in a well-defined statistical estimation problem. We denote the joint density of conditional network exposures $\mathbf{A}$ given $\mathbf{W}$ by $\mathbf{g}(\mathbf{A} \,|\, \mathbf{W})$, with above assumptions implying the factorization $\mathbf{g}(\mathbf{A} \,|\, \mathbf{W}) = \prod_{i=1}^N g(A_i \,|\, W_i^s)$. We denote the joint distribution of $\mathbf{W}$ by $\mathbf{Q_W}(\mathbf{W})$, making no additional assumptions of independence between $\mathbf{W} = (W_1, \ldots, W_N)$ and we assume $\mathbf{q_W}$ is a well-defined density for $\mathbf{Q_W}$, with respect to some dominating measure. This final set of assumptions defines our statistical model $\mathcal{M}$, where $\mathcal{M}$ describes the set of all possible distributions P^N for the observed dependent data $\mathbf{O}$.

We now introduce the notation $P = P_{\mathbf{Q},\mathbf{G}}$, for $\mathbf{Q} \equiv (\mathbf{Q_W}, Q_Y)$ and we assume the distributions $\mathbf{Q_W}$ and Q_Y are unspecified beyond the above modeling conditions **A1**, **A3**, **A4, B1** and **C1**. We also note that observed exposure model for $\mathbf{G}$ may be a restricted to incorporate the real-world knowledge about the true conditional treatment assignment, for example, when the common-in-i $g(A_i|W_i^s)$ is known, such as in a randomized clinical trial. This defines the statistical parametrization for the data-generating distribution of $\mathbf{O}$ in terms of the distributions $\mathbf{Q}$ and $\mathbf{G}$, and the corresponding statistical model is defined

as $\mathcal{M} = \{P_{\mathbf{Q},\mathbf{G}} : \mathbf{Q} \in \mathcal{Q}, \mathbf{G} \in \mathcal{G}\}$, where $\mathcal{Q}$ and $\mathcal{G}$ denote the parameter spaces for $\mathbf{Q}$ and $\mathbf{G}$, respectively. In particular, we denote $\mathbf{Q}_0$ as $\mathbf{Q}$ evaluated at P_0^N. Applying this newly introduced notation results in the likelihood:

$$p^N(\mathbf{O}) = \left[\prod_{i=1}^{N} q_Y(Y_i \,|\, A_i^s, W_i^s)\right] \left[\prod_{i=1}^{N} g(A_i \,|\, W_i^s)\right] \mathbf{q}_{\mathbf{W}}(\mathbf{W}). \tag{1}$$

We define an intervention of interest by replacing the conditional distribution $\mathbf{G}$ with a new user-supplied intervention $\mathbf{G}^*$ that has a density $\mathbf{g}^*$ that we assume is well-defined. Namely, $\mathbf{G}^*$ is a multivariate conditional distribution that encodes how each intervened exposure, denoted as A_i^*, is generated conditional on $\mathbf{W}$. We note that static or dynamic interventions on $\mathbf{A}$ correspond with degenerate choices of $\mathbf{g}^*$ (e.g., [102, 104, 105, 41, 136]), while non-degenerate choices of $\mathbf{g}^*$ are often referred to as stochastic interventions (e.g., [31, 106, 75, 137, 63]). We assume that $\mathbf{A}$ and $\mathbf{A}^*$ belong to the same common space $\mathcal{A}$ and we make no further restrictions on $\mathbf{G}^*$. We also define $A_i^{*s} := a_i^s(\mathbf{A}^*)$, where A_i^{*s} denotes the random variable implied by the summary measure $a_i^s(\cdot)$ mapping from an intervened exposure vector $\mathbf{A}^*$, for $i = 1, \ldots, N$. Finally, we define the post-intervention distribution $P_{\mathbf{Q},\mathbf{G}^*}$ by replacing $\mathbf{G}$ in $P_{\mathbf{Q},\mathbf{G}}$ with a new user-supplied distribution $\mathbf{G}^*$. We use $\mathbf{O}^* = (W_i, A_i^*, Y_i^*)_{i=1}^N$ to denote the random variable generated under $P_{\mathbf{Q},\mathbf{G}^*}$ and its likelihood is given by:

$$p^N_{\mathbf{Q},\mathbf{G}^*}(\mathbf{O}^*) = \left[\prod_{i=1}^{N} q_Y(Y_i^* \,|\, A_i^{*s}, W_i^s)\right] \mathbf{g}^*(\mathbf{A}^* \,|\, \mathbf{W}) \mathbf{q}_{\mathbf{W}}(\mathbf{W}). \tag{2}$$

The latter distribution $P_{\mathbf{Q},\mathbf{G}^*}$ is referred to as the G-computation formula for the post-intervention distribution of $\mathbf{O}$ under stochastic intervention $\mathbf{G}^*$ [103] and it is a parameter of P^N.

Our target statistical quantity ψ_0 is now defined as a function of this post-intervention distribution (2). Specifically, it is given by:

$$\psi_0 = \Psi(P_0^N) = E_{\mathbf{q}_0,\mathbf{g}^*}\left[\frac{1}{N}\sum_{i=1}^{N} Y_i^*\right],$$

which is an expectation of the sample-average of N outcomes among dependent units $i = 1, \ldots, N$, where the expectation is evaluated with respect to the post-intervention distribution $P_{\mathbf{Q},\mathbf{G}^*}$. We view $\Psi(P_0^N)$ as a mapping from the statistical model $\mathcal{M}$ into $\mathbb{R}$, and we note that ψ_0 is defined conditionally on the observed network structure, $\mathbf{F}$ and is also indexed by N. We also define $\bar{Q}(A_i^s, W_i^s) = \int_y y q_Y(y|\, A_i^s, W_i^s) d\mu(y)$ as the conditional mean evaluated under common-in-i distribution Q_Y, and $\bar{Q}_0$ as $\bar{Q}$ evaluated at P_0^N. Note that our dimension reduction assumptions imply that $E_{P_0^N}[Y_i \,|\, \mathbf{A}, \mathbf{W}] = \bar{Q}_0(A_i^s, W_i^s)$. We also note that our parameter ψ_0 only depends on P_0^N through $\bar{Q}_0$ and $\mathbf{Q}_{\mathbf{W},0}$, and with a slight abuse of notation we will interchangeably use $\Psi(P_0^N)$ and $\Psi(\bar{Q}_0, \mathbf{Q}_{\mathbf{W},0})$. Thus, the parameter ψ_0 is indexed

by N, $\mathbf{F}$ and $\mathbf{G}^*$ and can be written as:

$$\psi_0 = \frac{1}{N}\sum_{i=1}^{N}\int_{\mathbf{a},\mathbf{w}} \bar{Q}_0(a_i^s(\mathbf{a},\mathbf{w}), w_i^s(\mathbf{w}))\mathbf{g}^*(\mathbf{a}\,|\,\mathbf{w})\mathbf{q}_{\mathbf{W},0}(\mathbf{w})d\mu(\mathbf{a},\mathbf{w}),$$

with respect to some dominating measure $\mu(\mathbf{a},\mathbf{w})$.

One might also be interested in a target quantity defined as a contrast of two stochastic interventions. For example, one may define $\Psi^{\mathbf{G}_1^*}(P_0^N)$ and $\Psi^{\mathbf{G}_2^*}(P_0^N)$ as the above target parameter evaluated under stochastic interventions $\mathbf{G}_1^*$ and $\mathbf{G}_2^*$, respectively, then defining the target quantity as $\Psi^{\mathbf{G}_1^*,\mathbf{G}_2^*}(P_0^N) = \Psi^{\mathbf{G}_1^*}(P_0^N) - \Psi^{\mathbf{G}_2^*}(P_0^N)$. The average treatment effect over N connected units is then a special case of $\Psi^{\mathbf{G}_1^*,\mathbf{G}_2^*}(P_0^N)$ for interventions $\mathbf{G}_1^*,\mathbf{G}_2^*$ defined as $\mathbf{g}_1^*(\mathbf{1}^N\,|\,\mathbf{w}) = 1$ and $\mathbf{g}_2^*(\mathbf{0}^N\,|\,\mathbf{w}) = 1$, for any $\mathbf{w}\in\mathcal{W}$. We will focus on the estimation of the statistical parameter ψ_0 defined for one particular $\mathbf{G}^*$, noting that all of our results naturally generalize to contrasts or any other quantities that can be expressed as Euclidean-valued functions of a collection $\{\Psi^{\mathbf{G}^*}(P_0^N) : \mathbf{G}^*\in\mathcal{G}^*\}$, for a finite set of stochastic interventions $\mathcal{G}^*$.

We note that by making additional untestable assumptions, one can interpret ψ_0 as a causal quantity that measures the sample-average of the expected counterfactual outcomes in a network of N connected units under intervention $\mathbf{G}^*$, as was previously shown in [63]. However, these additional causal assumptions put no further restrictions on the above described probability distribution P_0^N, so that our statistical model $\mathcal{M}$ remains the same. Since $\mathcal{M}$ contains the true data distribution P_0^N, it follows that ψ_0 will always have a pure statistical interpretation as the feature $\Psi(P_0^N)$ of the data distribution P_0^N. For the estimation problem at hand, the causal model plays no further role: even when one does not believe any of the untestable causal assumptions, one might still argue that the statistical parameter ψ_0 represents an effect measure of interest controlling for all *measured* confounders. Finally, we note that the assumption **A1** can be dropped entirely, by defining the target parameter ψ_0 conditionally on the observed baseline covariates $\mathbf{W}$, as shown in [63].

1.3 Target parameter as a mapping applied to a mixture model

The above defined target parameter $\Psi(P_0^N)$ can be represented as an alternative (and equal) mapping $\bar{\Psi}(\bar{P}_0)$, where $\bar{P}_0$ is defined as a mixture of N unit-specific components of the joint data-generating distribution P_0^N. This leads us to another way of thinking about the estimation of our target parameter, suggesting that the problem of estimating ψ_0 should only be concerned with estimating the relevant components of the mixture $\bar{P}_0$. We first apply the summary measures $W_i^s = w_i^s(\mathbf{W})$ and $A_i^s = a_i^s(\mathbf{A},\mathbf{W})$ to the observed data $\mathbf{O}$, mapping it into a dataset of N dependent summary observations, denoted $\mathbf{O}^s = (O_1^s,\ldots,O_N^s)$ and referred to as the "*summary data*". We assume each $O_i^s = (W_i^s, A_i^s, Y_i)$ is distributed according to $P_{i,0}^s$, where $P_{i,0}^s$ is implied by the joint distribution P_0^N of $\mathbf{O}$ and the i-specific summary measures

$(w_i^s(\cdot), a_i^s(\cdot))$. We assume that each $P_{i,0}^s$ has a well-defined density $p_{i,0}^s$ with respect to some dominating measure. As before, the set of all possible distributions of $\mathbf{O}$ is given by the statistical model $\mathcal{M} = \{P_{\mathbf{Q},\mathbf{G}} : \mathbf{Q} \in \mathcal{Q}, \mathbf{G} \in \mathcal{G}\}$, and for a given $P^N \in \mathcal{M}$, we first define its implied mixture $\bar{P}$ and its relevant factors, and we then describe the new mapping $\bar{\Psi}(\bar{P})$ in Theorem 1.3.1. Our next goal is to present the efficient influence curve (EIC) for this mapping $\bar{\Psi}(\bar{P}_0)$, which we do in two steps in Theorems 1.3.2 and 1.3.3.

Mapping $\psi_0 = \bar{\Psi}(\bar{P}_0)$ for the mixture distribution $\bar{P}_0$.

Let $\mathcal{O}^s$ be the sigma-algebra for the union of the unit-specific supports of O_i^s, for $i = 1, \ldots, N$. For a set $A \in \mathcal{O}^s$, define the mixture distribution $\bar{P}(A)$ as a finite mixture of N unit-specific summary distributions P_i^s with constant weight $1/N$, i.e., $\bar{P} := 1/N \sum_{i=1}^N P_i^s$. Let $\bar{O}^s$ denote the random variable drawn from such $\bar{P}$ and we assume that $\bar{P}$ has a well-defined density $\bar{p} := 1/N \sum p_i^s$. Note that $\bar{p}$ can be factorized as follows:

$$\begin{aligned}\bar{p}(\bar{O}^s) &= \bar{p}(\bar{Y}, \bar{A}^s, \bar{W}^s) \\ &= \bar{q}_Y(\bar{Y} \mid \bar{A}^s, \bar{W}^s)\bar{g}(\bar{A}^s \mid \bar{W}^s)\bar{q}_W(\bar{W}^s),\end{aligned}$$

and we now describe in detail the above factors of $\bar{p}$.

First, let $Q_{W_i^s}$ denote the marginal distribution of the i-specific baseline summary measure W_i^s, with density $q_{W_i^s}(W_i^s)$ defined as the marginal of the joint density $p_i^s(Y_i, A_i^s, W_i^s)$. The distribution $\bar{Q}_W$ can then be defined as a finite mixture of these i-specific marginal distributions $Q_{W_i^s}$, and the density of $\bar{Q}_W$ can be defined as follows:

$$\bar{q}_W(w^s) := \frac{1}{N} \sum_{i=1}^N q_{W_i^s}(w^s).$$

We let $\bar{W}^s$ denote a random variable drawn from the mixture distribution $\bar{Q}_W$, noting that $\bar{W}^s$ belongs to the same common space $\mathcal{W}^s$ as all $w_i^s(\mathbf{W})$, for $i = 1, \ldots, N$. Similarly, we let H_i denote the i-specific joint distribution of the summaries (A_i^s, W_i^s), with its density $h_i(A_i^s, W_i^s)$ implied by $p_i^s(Y_i, A_i^s, W_i^s)$. We also let H_i^* denote the joint distribution of the summaries (A_i^{*s}, W_i^s), where A_i^{*s} is determined by the user-supplied stochastic intervention $\mathbf{G}^*_{\mathbf{A}^*|\mathbf{W}}$ and the i-specific summary measure a_i^s, and we denote the density of H_i^* as h_i^*. We also assume that these i-specific densities $h_i(a^s, w^s)$ and $h_i^*(a^s, w^s)$ are well-defined with respect to some common dominating measure $\mu_{a,w}$. We now define the mixture distribution $\bar{H}$ as a finite mixture of i-specific H_i, with its corresponding mixture density defined as $\bar{h}(a^s, w^s) := 1/N \sum_{i=1}^N h_i(a^s, w^s)$, and we let $(\bar{A}^s, \bar{W}^s)$ denote the random variables drawn jointly from $\bar{H}$. Next, we define an analogous mixture distribution $\bar{H}^*$ as a finite mixture of i-specific distributions H_i^*, with its mixture density given by $\bar{h}^*(a^s, w^s) := 1/N \sum_{i=1}^N h_i^*(a^s, w^s)$, and we let $(\bar{A}^{*s}, \bar{W}^s)$ denote the random variables drawn jointly from $\bar{H}^*$. Finally, we note that these mixture densities $\bar{h}$ and $\bar{h}^*$ can be factorized as follows:

$$\bar{h}(a^s, w^s) = \bar{g}(a^s|w^s)\bar{q}_W(w^s),$$

$$\bar{h}^*(a^s, w^s) = \bar{g}^*(a^s|w^s)\bar{q}_W(w^s),$$

where $\bar{g}$ is a factor in the above factorization of the likelihood of $\bar{O}^s$, namely, $\bar{g}$ is the density for the conditional distribution of $\bar{A}^s$ given $\bar{W}^s$, denoted as $\bar{G}$; $\bar{g}^*$ is the density for the conditional distribution of $\bar{A}^{*s}$ given $\bar{W}^s$, denoted as $\bar{G}^*$; and $\bar{q}_W$ is the previously defined marginal density for the mixture $\bar{Q}_W$. In a similar manner one can define the conditional distribution of $\bar{Y}$ given $(\bar{A}^s, \bar{W}^s)$, denoted as $\bar{Q}_Y$, with its density denoted as $\bar{q}_Y$, which completes the description of the three factors of $\bar{p}(\bar{O}^s)$. We also define a statistical model $\bar{\mathcal{M}}$ as the space of all possible distributions $\{\bar{Q}_Y, \bar{G}, \bar{Q}_W\}$ and we note that each $\bar{P} \in \bar{\mathcal{M}}$ is implied by some $P^N \in \mathcal{M}$. We also note that when $(\mathbf{W}, \mathbf{A})$ are discrete, one can obtain the following intuitive analytic expressions for the above defined densities $q_{W_i^s}$, h_i and h_i^*:

$$\begin{aligned}
q_{W_i^s}(w^s) &= \int_{\mathbf{w}} I(w_i^s(\mathbf{w}) = w^s)\mathbf{q_W}(\mathbf{w})d\mu_w(\mathbf{w}),\\
h_i(a^s, w^s) &= \int_{\mathbf{a},\mathbf{w}} I(a_i^s(\mathbf{a}, \mathbf{w}) = a^s, w_i^s(\mathbf{w}) = w^s)\mathbf{g}(\mathbf{a}|\mathbf{w})\mathbf{q_W}(\mathbf{w})d\mu_{a,w}(\mathbf{a}, \mathbf{w}),\\
h_i^*(a^s, w^s) &= \int_{\mathbf{a},\mathbf{w}} I(a_i^s(\mathbf{a}, \mathbf{w}) = a^s, w_i^s(\mathbf{w}) = w^s)\mathbf{g}^*(\mathbf{a}|\mathbf{w})\mathbf{q_W}(\mathbf{w})d\mu_{a,w}(\mathbf{a}, \mathbf{w}),
\end{aligned}$$

where μ_w and $\mu_{a,w}$ are some dominating measures. The new mapping $\psi_0 = \bar{\Psi}(\bar{P}_0)$ for our target parameter is now presented in the following theorem.

Theorem 1.3.1. *Let $P^N \in \mathcal{M}$ and let P_i^s denote the i-specific summary data distribution of $O_i^s = (W_i^s, A_i^s, Y_i)$. Let $\bar{P} \in \bar{\mathcal{M}}$ be the above defined finite mixture of these N unit-specific distributions P_i^s and $\bar{\mathcal{M}}$ is the above defined mixture model. Let $\bar{O}^s = (\bar{W}^s, \bar{A}^s, \bar{Y}) \sim \bar{P}$ denote one sample drawn from $\bar{P}$. The likelihood of $\bar{O}^s$ is given by:*

$$\bar{p}(\bar{O}^s) = q_Y(\bar{Y} \mid \bar{A}^s, \bar{W}^s)\bar{g}(\bar{A}^s \mid \bar{W}^s)\bar{q}_W(\bar{W}^s),$$

where $\bar{g}$ and $\bar{q}_W$ are the previously defined factors of $\bar{p}$; q_Y is the density of $Q_Y \in \mathcal{M}$ previously defined in Section 1.2, i.e., q_Y is the common-in-i conditional density of Y_i given (A_i^s, W_i^s). Due to the modeling assumptions on $\mathcal{M}$, q_Y is also the conditional density of $\bar{Y}$ given $(\bar{A}^s, \bar{W}^s)$. It follows that $\Psi(P^N) \equiv \bar{\Psi}(\bar{P})$, where the new mapping $\bar{\Psi}(\bar{P})$ is given by:

$$\begin{aligned}
\bar{\Psi}(\bar{Q}, \bar{Q}_W, \bar{g}^*) &= E_{\bar{Q}_W}\left[E_{\bar{g}^*}[\bar{Q}(\bar{A}^{*s}, \bar{W}^s) \mid \bar{W}^s]\right]\\
&= \int_{w^s \in \mathcal{W}^s, a^s \in \mathcal{A}^s} \bar{Q}(a^s, w^s)\bar{g}^*(a^s \mid w^s)d\bar{Q}_W(w^s)\\
&= \frac{1}{N}\sum_{i=1}^{N} E_{Q_{W_i^s}}\left[E_{\bar{g}^*}[\bar{Q}(\bar{A}^{*s}, W_i^s) \mid W_i^s]\right]\\
&= \frac{1}{N}\sum_{i=1}^{N} \int_{w_i^s, a^s} \bar{Q}(a^s, w_i^s)\bar{g}^*(a^s \mid w_i^s)dQ_{W_i^s}(w_i^s),
\end{aligned}$$

and we let $\bar{Q}(a^s, w^s) := E_{Q_Y}[\bar{Y} \mid \bar{A}^s = a^s, \bar{W}^s = w^s]$. With the slight abuse of notation we interchangeably write $\bar{\Psi}(\bar{Q}, \bar{Q}_W, \bar{g}^*)$ and $\bar{\Psi}(\bar{P})$, to emphasize the fact that $\bar{\Psi}$ depends on $\bar{P}$ only through $\bar{Q}$, $\bar{Q}_W$ and $\bar{g}^*$.

Proof. First, we show that that q_Y is indeed the conditional density of $\bar{Y}$, given $(\bar{A}^s, \bar{W}^s)$ under $\bar{P}$. To see this, note:

$$\begin{aligned}\bar{p}(w^s, a^s, y^s) &= \frac{1}{N}\sum_i p_i^s(w^s, a^s, y) \\ &= \frac{1}{N}\sum_i p_i^s(y|w^s, a^s) h_i(w^s, a^s) \\ &= q_Y(y|w^s, a^s)\frac{1}{N}\sum_{i=1}^{N} h_i(w^s, a^s),\end{aligned}$$

where by the assumptions in Section 1.2 we note that the distribution $P(Y_i|W_i^s, A_i^s)$ is given by a common distribution Q_Y with density q_Y, from which the above result follows as claimed. The equivalence $\Psi(P^N) \equiv \bar{\Psi}(\bar{P})$ then follows directly by applying the definitions of $\bar{g}^*$, $\bar{Q}_W$ and $\bar{Q}$. □

The above theorem implies that the estimator of ψ_0 can be obtained from the estimators of $\bar{Q}_0$ and $\bar{Q}_{W,0}$. It also gives us an alternative interpretation for our target parameter ψ_0. Namely, the mapping $\bar{\Psi}(\bar{P}_0)$ happens to be equal to the parameter given by the G-computation formula for the mean outcome $E\bar{Y}_{\bar{g}_0^*}$, under stochastic intervention $\bar{g}_0^*$ and the observed data $(\bar{W}^s, \bar{A}^s, \bar{Y}) \sim \bar{P}_0$. That is, we take the above defined conditional density $\bar{g}_0^* := \bar{h}^*(\mathbf{G}^*, \mathbf{Q}_{\mathbf{W},0})/\bar{q}_W(\mathbf{Q}_{\mathbf{W},0})$ as if this was a known user-supplied stochastic intervention on $\bar{A}^s$ given $\bar{W}^s$, treating $\bar{g}_0^*$ as fixed we then evaluate $E\bar{Y}_{\bar{g}_0^*}$ by first replacing $\bar{g}_0(\bar{A}^s|\bar{W}^s)$ factor of $\bar{p}(\bar{O}^s)$ with $\bar{g}_0^*$, then taking the expectation of $\bar{Y}$ under such modified post-intervention distribution $\bar{P}^*$. We also recognize that $\bar{g}_0^*$ will be generally unknown, since it depends on the true distribution of the data via $Q_{\mathbf{W},0}$. Nonetheless, expressing the dependent-data parameter as some function of the mixture $\bar{P}_0$ implies that the estimation of this parameter can be accomplished by simply treating the observed dependent units as if they are independent and identically distributed (iid) (see Lemma 1.4.1 from Section 1.4 for a specific case of estimating $\bar{g}_0$ component of mixture $\bar{p}$). Hence, whenever we are concerned with estimating any parameter of $\bar{P}_0$, such as ψ_0 given above, we can ignore the dependence among units O_i^s, $i = 1, \ldots, N$, immediately providing us with an iid-analogue estimator for ψ_0, and in our case we will undertake the iid targeted maximum likelihood estimation (TMLE) approach, as described in Section 1.4. Among a class of iid-analogue estimators for ψ_0, we can choose an estimator which would be efficient for iid data, and in our case, we will show that such an analogous efficient iid TMLE will also be semi-parametrically efficient in our dependent data model.

The efficient influence curve

The efficient influence curve (EIC), frequently referred to as the efficient score or canonical gradient, is a key ingredient in semi-parametric efficient estimation, because it defines

the linear approximation of any efficient and regular asymptotically linear estimator, and therefore provides an asymptotic bound for the variance of all regular asymptotically linear estimators [12]. Furthermore, as discussed in Section 5 of [63], even for dependent data problems such as ours, the EIC still characterizes the limiting normal distribution of the MLE, thus establishing that if we want to construct an estimator that is asymptotically equivalent to the MLE, we need to study the EIC of our target parameter. Due to local asymptotic normality of the log-likelihood, as was argued in [123], the normal limiting distribution implied by the MLE is still the optimal limit distribution in the convolution theorem for efficient estimators. Our first result provides the EIC $\bar{D}^N$ for a data-adaptive parameter $\bar{\Psi}_N(\bar{Q}_0, \bar{Q}_{W,0}) := \bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0}, \bar{g}_N^*)$ indexed by fixed $\bar{g}^* = \bar{g}_N^*$, where $\bar{g}_N^*$ is an estimator of the true density $\bar{g}_0^*$. We will refer to $\bar{D}^N$ as the iid-EIC, since it is a direct analogue of the iid-data EIC for the parameter $E\bar{Y}_{\bar{g}^*}$. We then present the EIC $\bar{D}$ for our actual parameter of interest $\bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0}, \bar{g}_0^*)$, defined with respect to the true density $\bar{g}_0^*$.

EIC for data-adaptive parameter indexed by fixed stochastic intervention $\bar{g}_N^*$

We now replace $\bar{g}_0^*$ by a fixed density $\bar{g}^*$, set equal to some data-dependent estimator $\bar{g}_N^*$ of $\bar{g}_0^*$, which is then treated as fixed. This allows us to define the data-adaptive target parameter $\bar{\Psi}_N(\bar{Q}_0, \bar{Q}_{W,0})$, indexed by such fixed $\bar{g}^*$, as $E_{\bar{Q}_{W,0}}\left[E_{\bar{g}^*=\bar{g}_N^*}[\bar{Q}_0(\bar{A}^s, \bar{W}^s) \,|\, \bar{W}^s]\right]$. From iid results for parameters defined under fixed stochastic interventions, such as those described in [75], it immediately follows that the efficient influence curve for parameter $\bar{\Psi}_N(\bar{P}_0)$ at $\bar{P}_0 \in \bar{\mathcal{M}}$ and $\bar{O}^s \sim \bar{P}_0$ is given by:

$$\bar{D}^{IID}(\bar{P}_0)(\bar{O}^s) = \frac{\bar{g}^*}{\bar{g}_0}(\bar{A}^s \,|\, \bar{W}^s)\left[\bar{Y} - \bar{Q}_0(\bar{A}^s, \bar{W}^s)\right] + \left[E_{\bar{g}^*=\bar{g}_N^*}[\bar{Q}_0(\bar{A}^s, \bar{W}^s) \,|\, \bar{W}^s] - \bar{\Psi}_N(\bar{Q}_0, \bar{Q}_{W,0})\right].$$

In other words, we have obtained an efficient influence curve for the mean outcome of $\bar{Y}$ under stochastic intervention $\bar{g}^*$, for one observation $\bar{O}^s \sim \bar{P}_0$. We will thus refer to $\bar{D}^{IID}(\bar{P}_0)(\bar{O}^s)$ as the iid-EIC, due to just described iid-data interpretation of parameter $\bar{\Psi}_N(\bar{P})$. However, we don't get to observe $\bar{O}^s$, instead, our observed data is $\mathbf{O}^s = (O_1^s, \ldots, O_N^s)$, where $O_i^s \sim P_{i,0}^s$. Nonetheless, it follows that the EIC for the actual data-adaptive parameter $\bar{\Psi}_N(\bar{P}_0)$ at $P_0^N \in \mathcal{M}$ and the observed data model $\mathbf{O}^s \sim P_0^N$ is given by the sum of these iid-EICs, evaluated at i-specific observations O_i^s and scaled by $1/N$, i.e, the EIC for parameter $\bar{\Psi}_N(\bar{P}_0)$ is given by $1/N \sum_{i=1}^N \bar{D}^{IID}(\bar{P}_0)(O_i^s)$ and we also present this EIC in the following theorem.

Theorem 1.3.2. *Suppose our parameter of interest is defined by the mapping $\bar{\Psi}_N(\bar{P})$, where $\bar{P} \in \bar{\mathcal{M}}$ and $\bar{g}^*$ is fixed. The efficient influence curve $\bar{D}^N(P^N)(\mathbf{O}^s)$ for $\bar{\Psi}_N(\bar{P})$, evaluated at $P^N \in \mathcal{M}$ and one observation $\mathbf{O}^s$ (consisting of N dependent units) is given by*

$$\begin{aligned}\bar{D}^N(P^N)(\mathbf{O}^s) =& \frac{1}{N}\sum_{i=1}^N \left(\left[\frac{\bar{g}^*}{\bar{g}}(A_i^s \,|\, W_i^s)\left(Y_i - \bar{Q}(A_i^s, W_i^s)\right)\right] + \left[E_{\bar{g}^*}[\bar{Q}(A_i^s, W_i^s) \,|\, W_i^s] - \bar{\Psi}_N(\bar{Q}, \bar{Q}_W)\right] \right) \\ =& \frac{1}{N}\sum_{i=1}^N \left(\bar{D}_{Y_i}(\bar{Q}, \bar{g})(O_i^s) + \bar{D}_{W_i^s}(\bar{Q}, \bar{Q}_W)(W_i^s)\right),\end{aligned}$$

where

$$\begin{aligned}\bar{D}_{W_i^s}(\bar{Q},\bar{Q}_W)(W_i^s) &= E_{\bar{g}^*}[\bar{Q}(A_i^s,W_i^s)\,|\,W_i^s] - \bar{\Psi}_N(\bar{Q},\bar{Q}_W) \\ &= \int_{a^s} \bar{Q}(a^s,W_i^s)\bar{g}^*(a^s|W_i^s) - \frac{1}{N}\sum_{i=1}^{N}\int_{a^s,w^s} \bar{Q}(a^s,w_i^s)\bar{g}^*(a^s|w_i^s)Q_{W_i^s}(w^s).\end{aligned}$$

Proof. See Appendix B of our technical report [114]. □

EIC for parameter under true $\bar{g}^*(\mathbf{g}^*, Q_{\mathbf{W},0})$.

We now consider our actual target parameter $\bar{\Psi}(\bar{P}_0) := \bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0}, \bar{g}_0^*)$, obtained by replacing fixed $\bar{g}^* := \bar{g}_N^*$ in $\Psi_N(\bar{Q}_0, \bar{Q}_{W,0})$ with true density $\bar{g}_0^* := \bar{h}(\mathbf{G}^*, \mathbf{Q}_{\mathbf{W},0})/\bar{q}_W(\mathbf{Q}_{\mathbf{W},0})$. As a result, the EIC for the parameter $\bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0}, \bar{g}_0^*)$ will contain an additional contributing term $D^{\bar{g}^*}(P_0^N)(\mathbf{W})$ due to $\Psi_N(\bar{Q}_0, \bar{Q}_{W,0}) - \Psi(\bar{Q}_0, \bar{Q}_{W,0}, \bar{g}_0^*)$. This additional contribution is derived in Appendix C of our technical report [114]. The final EIC for our actual parameter $\bar{\Psi}(\bar{P})$ is given by $\bar{D}(P^N)(\mathbf{O}) = \bar{D}^N(P_0^N)(\mathbf{O}^s) + \bar{D}^{\bar{g}^*}(P_0^N)(\mathbf{W})$ and is provided in Theorem 1.3.3 below. We note that the resulting EIC provided here is to same estimating function proposed in Section 11 of technical report [64] for estimation of ψ_0 in a model that does not assume independence of $\mathbf{W} = (W_1, \ldots, W_N)$.

Theorem 1.3.3. *Suppose our parameter is given by the mapping $\bar{\Psi}(\bar{P})$ defined in Section 1.3. The efficient influence curve for $\bar{\Psi}(\bar{P})$ is given by:*

$$\begin{aligned}\bar{D}(P^N)(\mathbf{O}) =& \bar{D}^N(P^N)(\mathbf{O}^s) + \bar{D}^{\bar{g}^*}(P^N)(\mathbf{W}) \\ =& \frac{1}{N}\sum_{i=1}^{N}\left(\left[\frac{\bar{g}^*}{\bar{g}}(A_i^s\,|\,W_i^s)\left(Y_i - \bar{Q}(A_i^s,W_i^s)\right)\right] + \left[\int_{a^s}\bar{Q}(a^s,W_i^s)\bar{g}^*(a^s|W_i^s) - \bar{\Psi}(\bar{Q},\bar{Q}_W)\right]\right) \\ &+ \frac{1}{N}\sum_{i=1}^{N}\left[\int_{a^s}\bar{Q}(a^s,W_i^s)g_i^*(a^s|\mathbf{W}) - \int_{a^s}\bar{Q}(a^s,W_i^s)\bar{g}^*(a^s|W_i^s)\right],\end{aligned}$$

where

$$g_i^*(a^s|\mathbf{W}) = \int_{\mathbf{a}} I(a_i^s(\mathbf{a},\mathbf{W}) = a^s)\mathbf{g}^*(\mathbf{a}|\mathbf{W})d\mu_{\mathbf{a}}(\mathbf{a}),$$

$$\bar{\Psi}(\bar{Q},\bar{Q}_W) = \frac{1}{N}\sum_{i=1}^{N}\int_{a^s,w^s}\bar{Q}(a^s,w_i^s)\bar{g}^*(a^s|w_i^s)Q_{W_i^s}(w^s),$$

and $\bar{D}(P^N)$ above can be further simplified as:

$$\begin{aligned}\bar{D}(P^N)(\mathbf{O}) &= \frac{1}{N}\sum_{i=1}^{N}\left(\left[\frac{\bar{g}^*}{\bar{g}}(A_i^s\,|\,W_i^s)\left(Y_i - \bar{Q}(A_i^s,W_i^s)\right)\right] + \left[\int_{a^s}\bar{Q}(a^s,W_i^s)g_i^*(a^s|\mathbf{W}) - \bar{\Psi}(\bar{Q},\bar{Q}_W)\right]\right) \\ &= \frac{1}{N}\sum_{i=1}^{N}\left(\bar{D}_{Y_i}(\bar{Q},\bar{g})(O_i^s) + \bar{D}_{g_i^*}(\bar{Q},\bar{Q}_W)(\mathbf{W})\right).\end{aligned}$$

Proof. See Appendix C of our technical report [114]. □

Suppose that $\bar{g}^*/\bar{g}$ is uniformly bounded on a set that contains (W_i^s, A_i^s) with probability 1, for all i. Using similar analysis to the one conducted in [63], we can show that $\bar{D}$ above is a doubly robust estimating function for parameter $\psi_0 = \bar{\Psi}(\bar{P}_0)$, in the sense that,

$$P_0\bar{D}(\bar{Q}, \bar{g}_0, \psi_0) = P_0\bar{D}(\bar{Q}_0, \bar{g}, \psi_0) = P_0\bar{D}(\bar{Q}_0, \bar{g}_0, \psi_0) = 0,$$

where $P_0 f = \int f(o) dP_0^N(o)$ denotes the expectation of f under distribution P_0^N, and $\bar{Q} = (\bar{Q}, \bar{Q}_W)$. This implies that any estimator that solves this equation is going to be consistent if: (1) $\bar{Q}_{W,N}$ is consistent for $\bar{Q}_{W,0}$ and (2) at least one of the two estimators $\bar{Q}_N$ or $\bar{g}_N$ is consistent for $\bar{Q}_0$ or $\bar{g}_0$.

1.4 The Targeted Maximum Likelihood Estimation (TMLE)

We had found a new representation for our target parameter $\Psi(P_0^N) = \bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0})$, which shows that our parameter ψ_0 depends on P_0^N only as a function of its mixture, $\bar{P}_0$, and in particular, its a function of $\bar{Q}_Y$ and $\bar{Q}_W$. Demonstrating that our parameter can be written as a mapping $\bar{\Psi}(\bar{Q}, \bar{Q}_W)$ is hence the first step in estimation of ψ_0. It implies that the estimation of ψ_0 should now only be concerned with estimating the relevant factors of $\bar{P}_0$ and we proceed by following the usual Targeted Maximum Likelihood Estimation (TMLE) template. For the description of the TMLE framework in iid data with static interventions, we refer to [67, 49, 66], and for the TMLE in iid data with stochastic intervention, we refer to [75].

We note that our TMLE for estimating ψ_0 will be described in terms of the iid estimators of the relevant factors of $\bar{P}_0$, namely, the estimators $\bar{Q}_N$, $\bar{g}_N$, $\bar{g}_N^*$ and $\bar{Q}_{W,N}$ of $\bar{Q}_0$, $\bar{g}_0$, $\bar{g}_0^*$ and $\bar{Q}_{W,0}$, respectively. Our next step is then to create a targeted estimator $\bar{Q}_N^*$ of $\bar{Q}_0$ by updating the initial estimator $\bar{Q}_N$, defining the TMLE ψ_N^* as the corresponding plug-in estimator for the mixture mapping $\bar{\Psi}(\bar{Q}_N^*, \bar{Q}_{W,N})$. We define the targeted update $\bar{Q}_N^*$ based on the loss function for $\bar{Q}_0$ and the least favorable fluctuation submodel through $\bar{Q}_0$ in terms of $\bar{g}_0$ and $\bar{g}_0^*$. The model update $\bar{Q}_N^*$ is defined in a way so that its score generates the efficient influence curve $\bar{D}$ presented in Theorem 1.3.3. That is, the targeted estimator $\bar{Q}_N^*$ updates $\bar{Q}_N$ by: (1) using the estimated weights $\bar{g}_N^*/\bar{g}_N$, (2) using a parametric submodel $\{\bar{Q}_N(\varepsilon, \bar{g}_N^*/\bar{g}_N)\}$ through the initial estimator $\bar{Q}_N = \bar{Q}_N(0, \bar{g}_N^*/\bar{g}_N)$ at $\varepsilon = 0$, where $\{\bar{Q}_N(\varepsilon, \bar{g}_N^*/\bar{g}_N)\}$ is referred to as the least-favorable submodel, (3) fitting ε with the standard parametric MLE, with ε^N denoting this fit, and finally, (4) defining the targeted (updated) estimator as $\bar{Q}_N^* := \bar{Q}_N(\varepsilon^N, \bar{g}_N^*/\bar{g}_N)$. The TMLE ψ_N^* of ψ_0 is then defined as the corresponding substitution estimator $\psi_N^* = \bar{\Psi}(\bar{Q}_N^*, \bar{Q}_{W,N})$. We also note that this TMLE is actually the usual iid TMLE algorithm for estimating the quantity $EY_{\bar{g}^*}$ under fixed (data-adaptive) $\bar{g}^*$, treating observations O_i^s, for $i = 1, \ldots, N$ as if they are iid. Finally, we note

that the TMLE we present here is a semi-parametrically efficient estimator for ψ_0, since its algebraically equivalent to the TMLE presented in [63], as we discuss in more detail in Appendix E of our technical report [114]. Thus, the TMLE ψ_N^* solves the empirical score equation given by the efficient influence curve $\bar{D}$ from Theorem 1.3.3, implying that ψ_N^* also inherits the double robustness property of this efficient influence curve.

The estimator $\bar{Q}_{W,N}$ for $\bar{Q}_{W,0}$

We define an estimator $\bar{Q}_{W,N}$ of $\bar{Q}_{W,0}$ by first defining the empirical counterpart $\mathbf{Q}_{\mathbf{W},N}$ of $\mathbf{Q}_{\mathbf{W},0}$ that puts mass one on the observed $\mathbf{W} = (W_1, \ldots, W_N)$, which then implies that the empirical distribution $Q_{W_i^s,N}$ of $Q_{W_i^s,0}$ will put mass one on its corresponding observed $W_i^s = w_i^s(\mathbf{W})$, for $i = 1, \ldots, N$. Hence, for each $w^s \in \mathcal{W}^s$, the empirical counterpart $\bar{Q}_{W,N}(w^s)$ of $\bar{Q}_{W,0}(w^s)$ may be defined as follows:

$$\bar{Q}_{W,N}(w^s) := \frac{1}{N} \sum_{i=1}^{N} I(W_i^s = w^s).$$

The initial (non-targeted) estimator $\bar{Q}_N$ of $\bar{Q}_0$

We assumed there is a common model $\bar{Q}_0$ across all i and Y_i are conditionally independent given (A_i^s, W_i^s), for all i. Consequently, the estimation of a common $\bar{Q}_N$ can proceed by using the pooled summary data (W_i^s, A_i^s, Y_i), $i = 1, \ldots, N$, as if the sample is iid across i and one can rely on the usual parametric MLE or loss-based cross-validation for estimating $\bar{Q}_N$, as described in [63]. Given that Y_i can be continuous or discrete for some known range $Y_i \in [a, b]$, for $i = 1, \ldots, N$, the estimation of $\bar{Q}_0$ can be based on the following log-likelihood loss,

$$L(\bar{Q})(Y \mid A^s, W^s) = -\sum_{i=1}^{N} \log \left\{ \bar{Q}(A_i^s, W_i^s)^{Y_i} (1 - \bar{Q}(A_i^s, W_i^s))^{1-Y_i} \right\},$$

or the squared error loss

$$L(\bar{Q})(O^s) = -\sum_{i=1}^{N} \left(Y_i - \bar{Q}(A_i^s, W_i^s) \right)^2.$$

Thus, fitting $\bar{Q}_N$ for common $\bar{Q}_0 = E[Y_i \mid A_i^s, W_i^s]$ amounts to using the summary data structure (W_i^s, A_i^s, Y_i), for $i = 1, \ldots, N$. In other words, we use the entire sample of N observations for predicting Y_i. For example, for binary Y_i, $\bar{Q}_N$ can be estimated by fitting a single logistic regression model to all N observations, with Y_i as the outcome, (W_i^s, A_i^s) as predictors, and possibly adding the number of friends, $|F_i|$, as an additional covariate. After fitting $\bar{Q}_N$, one generates a vector of unit-specific prediction values, $(\bar{Q}_N(A_i^s, W_i^s))_{i=1}^N$, which are then used to build an updated version $\bar{Q}_N^*$ of $\bar{Q}_N$.

Estimating $\bar{g}_0^*$ and $\bar{g}_0$

We now describe a direct approach to estimation of $\bar{g}_0$ that relies on Lemma 1.4.1 below. This lemma states that a consistent estimator $\bar{g}_N$ of $\bar{g}_0$ can be obtained by taking a pooled sample (A_i^s, W_i^s), for $i = 1, \ldots, N$, and using the usual iid maximum likelihood-based estimation, as if we were fitting a common-in-i conditional density for A_i^s given W_i^s and treating (A_i^s, W_i^s) as independent observations. For example, if each component of A_i^s is binary, and $|A_i^s| = k$ for all i, the conditional distribution for $\bar{g}_0$ could be factorized in terms of the product of k binary conditional distributions. Each of these binary conditional distributions can be estimated with the usual logistic regression methods. We also refer to Section 1.6 for a running example that describes in detail this direct estimation approach. Suppose now that $\mathbf{g}_0$ is known, as will be the case in a randomized clinical trial (RCT). We note that this aforementioned approach to estimating $\bar{g}_0$ can be easily adopted to incorporate the knowledge of true $\mathbf{g}_0$. That is, one could proceed by first simulating a very large number of observations $(A_j^s, W_j^s)_{j=1}^M$ from $(\mathbf{g}_0, \mathbf{Q}_{\mathbf{W},N})$, with $\mathbf{Q}_{\mathbf{W},N}$ that puts mass one on the observed $\mathbf{W}$, and then fitting the maximum likelihood-based estimator for $\bar{g}_0$, as if we were fitting a common model for A_i^s given W_i^s, based on this very large sample that is treated as iid.

As discussed in the previous section, $\bar{g}_0^* := \bar{h}^*(\mathbf{G}^*, \mathbf{Q}_{\mathbf{W},0})/\bar{q}_W(\mathbf{Q}_{\mathbf{W},0})$ will generally be unknown and hence will also need to be estimated from the data, in particular, since $\bar{g}_0^*$ depends on the true distribution of the data via $\mathbf{Q}_{\mathbf{W},0}$. Therefore, we propose estimating $\bar{g}_0^*$ by using the same method as for for estimating $\bar{g}_0$ in case when $\mathbf{g}_0$ is known. Namely, we propose replacing known $\mathbf{g}_0$ with known $\mathbf{g}^*$ and then simulating a very large number of observations $(A_j^{*s}, W_j^s)_{j=1}^M$ from $(\mathbf{g}^*, \mathbf{Q}_{\mathbf{W},N})$, then using the same maximum likelihood-based approach to obtain an estimator $\bar{g}_N^*$ of $\bar{g}_0^*$, treating this simulated sample as if iid. Finally, even when $\mathbf{g}_0$ is unknown, such as in an observational study on N connected units, one could obtain a better estimator of $\bar{g}_0$ by utilizing the conditional independence assumptions for observed exposures A_i, given W_i^s, for $i = 1, \ldots, N$. Similar to estimation of $\bar{Q}_N$, this allows us to use loss-based cross-validation and machine learning methods to obtain a good approximation $g_N(a|w^s)$ for common-in-i density $g_0(a|w^s)$, resulting in an estimator $\mathbf{g}_N$ of the joint density $\mathbf{g}_0$. We can now repeat the above described procedure for estimating $\bar{g}_0$ when $\mathbf{g}_0$ is known, except using such data-adaptively estimated $\mathbf{g}_N$ instead of $\mathbf{g}_0$. In this manner, one can obtain sufficient approximations to true $\bar{g}_0$ and $\bar{g}_0^*$, by fully utilizing the actual model knowledge for $\mathbf{g}_0$ and the actual knowledge of $\mathbf{g}^*$. Finally, we use these fits to evaluate $\bar{g}_N^*/\bar{g}_N$ at each observed (A_i^s, W_i^s), yielding N predictions $(\bar{g}_N^*(A_i^s \,|\, W_i^s)/\bar{g}_N(A_i^s \,|\, W_i^s))$, for $i = 1, \ldots, N$, which will then be used as unit-level weights during the TMLE modeling update of the estimator $\bar{Q}_N$.

Lemma 1.4.1. *(Lemma 2 in [63]). Let the density $\bar{g}_0$ be defined as*

$$\bar{g}_0(\bar{a}^s \,|\, \bar{w}^s) := \bar{h}_0(\bar{a}^s, \bar{w}^s)/\bar{q}_{W,0}(\bar{w}^s),$$

where $\bar{q}_{W,0}$ and $\bar{h}_0$ are as previously defined in Section 1.3. Let $\mathcal{H}^g$ be a set of functions that

contain true $\bar{g}_0$, then

$$\bar{g}_0 = \arg\max_{\bar{g}\in\mathcal{H}^g} E_{P_0^N}\left\{\frac{1}{N}\sum_i \log \bar{g}(A_i^s \mid W_i^s)\right\}.$$

A consistent estimator $\bar{g}_N$ for $\bar{g}_0$ can be obtained by plugging in the empirical counterpart of P_0^N above, resulting in an estimator:

$$\bar{g}_N = \arg\max_{\bar{g}\in\mathcal{H}^g} \left\{\frac{1}{N}\sum_{i=1}^{N} \log \bar{g}(A_i^s \mid W_i^s)\right\}.$$

That is, $\bar{g}_N$ is the maximum likelihood estimator of $\bar{g}_0$ that uses the pooled sample (A_i^s, W_i^s), for $i = 1,\ldots,N$, treating the dependent N units as iid.

Proof. See Appendix A of our technical report [114]. □

The TMLE algorithm ψ_N^* for N connected units

Having defined the estimators $\bar{Q}_N$, $\bar{g}_N$, $\bar{g}_N^*$ and $\bar{Q}_{W,N}$, the TMLE ψ_N^* is obtained by first constructing the model update $\bar{Q}_N^*$ for $\bar{Q}_N$, as described in step 1. below, and then evaluating ψ_N^* as a substitution estimator for the mapping $\bar{\Psi}$, as described in step 2. below.

1. Define the following parametric submodel for $\bar{Q}_N$: $\text{Logit}\bar{Q}_N(\varepsilon) = \varepsilon + \text{Logit}\bar{Q}_N$ and define the following weighted log-likelihood loss function:

$$L^w(\bar{Q}_N(\varepsilon))(\mathbf{O}^s) = -\sum_{i=1}^{N} \log\left\{\bar{Q}_N(\varepsilon)(A_i^s, W_i^s)^{Y_i}(1-\bar{Q}_N(\varepsilon)(A_i^s, W_i^s))^{1-Y_i}\right\}\frac{\bar{g}_N^*}{\bar{g}_N}(A_i^s \mid W_i^s).$$

The model update $\bar{Q}_N^*$ is defined as $\bar{Q}_N(\varepsilon^N) = \text{Expit}\left(\text{Logit}\bar{Q}_N + \varepsilon^N\right)$, where ε^N minimizes the above loss, i.e., $\varepsilon^N = \arg\min_\varepsilon L^w(\bar{Q}_N(\varepsilon))(\mathbf{O}^s)$. That is, one can fit ε^N by simply running the intercept-only weighted logistic regression using the pooled sample of N observations (W_i^s, A_i^s, Y_i), for $i = 1,\ldots,N$, with outcome Y_i, intercept ε, using offsets $\text{Logit}\bar{Q}_N(A_i^s, W_i^s)$, predicted weights $\bar{g}_N^*(A_i^s \mid W_i^s)/\bar{g}_N(A_i^s \mid W_i^s)$ and no covariates. The fitted intercept is the maximum likelihood fit ε^N for ε, yielding the model update $\bar{Q}_N^*$,which can be evaluated for any fixed (a^s, w^s), by first computing the initial model prediction $\bar{Q}_N(a^s, w^s)$ and then evaluating the update $\bar{Q}_N(\varepsilon^N)$.

2. The TMLE $\psi_N^* = \bar{\Psi}_N(\bar{Q}_N^*, \bar{Q}_{W,N})$ of ψ_0 is defined as the following substitution estimator:

$$\psi_N^* = \frac{1}{N}\sum_{i=1}^{N}\int_{a^s} \bar{Q}_N^*(a^s, W_i^s)\bar{g}_{N,NPMLE}^*(a^s \mid W_i^s)d\mu(a^s),$$

where $\bar{g}_{N,NPMLE}^*$ is a NPMLE substitution estimator for $\bar{g}_0^* := \bar{h}^*(\mathbf{G}^*, \mathbf{Q}_{\mathbf{W},0})/\bar{q}_W(\mathbf{Q}_{\mathbf{W},0})$, obtained by plugging in the user-defined $\mathbf{G}^*$ and the empirical counterpart $\mathbf{Q}_{\mathbf{W},N}$ for

$\mathbf{Q}_{\mathbf{W},0}$ which puts mass one on observed $\mathbf{W} = (W_1, \ldots, W_N)$. Hence, the estimator $\bar{g}^*_{N,NPMLE}$ is defined as follows:

$$\bar{g}^*_{N,NPMLE}(a^s|w^s) \quad = \quad \frac{1/N \sum_i h_i(\mathbf{G}^*, \mathbf{Q}_{\mathbf{W},N})(a^s, w^s)}{1/N \sum_i Q_{W_i^s,N}(w^s)},$$

for each $(a^s, w^s) \in (\mathcal{A}^s, \mathcal{W}^s)$. The above TMLE ψ^*_N might require evaluation of $\bar{g}^*_{N,NPMLE}$ for every possible $a^s(\mathbf{a}, \mathbf{W})$ in the support of $\mathbf{A}^*$, and hence could be computationally challenging to implement in practice, especially for non-degenerate $\mathbf{g}^*$ and multivariate $(a^s_i : i = 1, \ldots, N)$. However, we also note that the above TMLE ψ^*_N takes on the following algebraically equivalent form:

$$\psi^*_N = \frac{1}{N} \sum_{i=1}^{N} \int_{\mathbf{a}} \bar{Q}^*_N(a^s_i(\mathbf{a}, \mathbf{W}), w^s_i(\mathbf{W})) \mathbf{g}^*(\mathbf{a} \,|\, \mathbf{W}) d\mu(\mathbf{a}),$$

which does not require computing $\bar{g}^*_{N,NPMLE}$. For non-degenerate $\mathbf{g}^*$, the latter expression for ψ^*_N can be closely approximated by sampling from $\mathbf{g}^*$ and performing Monte Carlo integration. That is, we propose evaluating ψ^*_N by iterating the following procedure $j = 1, \ldots, M$ times: (1) Sample N observations $\mathbf{A}^*_j = (A^*_{j,1}, \ldots, A^*_{j,N})$ from $\mathbf{g}^*(\mathbf{a}|\mathbf{W})$, conditionally on observed $\mathbf{W} = (W_1, \ldots, W_N)$; (2) Apply the summary measure mappings, constructing the following summary dataset $(A^{*s}_{j,i}, W^s_i)$, for $i = 1, \ldots, N$, where each $A^{*s}_{j,i} := a^s_i(\mathbf{A}^*_j, \mathbf{W})$; and (3) Evaluate the Monte Carlo approximation to ψ^*_N for iteration j as:

$$\psi^*_{j,N} = \frac{1}{N} \sum_{i=1}^{N} \bar{Q}^*_N(A^{s*}_{j,i}, W^s_i).$$

The Monte Carlo estimate $\bar{\psi}^*_N$ of ψ^*_N is then obtained by averaging $\psi^*_{j,N}$ across $j = 1, \ldots, M$, where M is chosen large enough to guarantee a small approximation error to ψ^*_N. Finally, we note that one could substantially reduce the computation time of this algorithm by simply re-using the summary datasets $(A^{*s}_{j,i}, W^s_i)$ that were already constructed while performing direct estimation of $\bar{g}^*_0$ from Section 1.4.

1.5 Asymptotic normality and inference for the TMLE

Having defined the TMLE $\psi^*_N = \bar{\Psi}(\bar{Q}^*_N, \bar{Q}_{W,N})$ for our parameter $\bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0})$, our goal now is to conduct inference. However, we start with an asymptotic analysis of the process $(\psi^*_N - \bar{\Psi}_N(\bar{Q}_0, \bar{Q}_{W,0}))$, where $\bar{\Psi}_N(\bar{Q}_0, \bar{Q}_{W,0})$ is a data-adaptive target parameter indexed by fixed $\bar{g}^*_N$. We then show that our results can be easily extended to allow inference for our original parameter of interest $\bar{\Psi}(\bar{Q}_0, \bar{Q}_{W,0})$ defined with respect to true $\bar{g}^*_0$.

As described in the Appendix D of our technical report [114], TMLE $\bar{\Psi}(\bar{Q}_N^*, \bar{Q}_{W,N})$ is constructed to solve the following empirical score equation:

$$\frac{1}{N}\sum_{i=1}^{N} \bar{D}^N(\bar{Q}_N^*, \bar{Q}_{W,N}, \bar{g}_N)(O_i^s) = 0,$$

where $\bar{D}^N(\bar{Q}, \bar{g})$ is the EIC for the data-adaptive parameter $\bar{\Psi}_N(\bar{Q}) := E\bar{Y}_{\bar{g}^*=\bar{g}_N^*}$ (Theorem 1.3.2). Using the identity for $\bar{D}^N(\bar{Q}, \bar{g})$ shown in the Appendix D of [114], we have that:

$$\bar{\Psi}_N(\bar{Q}) - \bar{\Psi}_N(\bar{Q}_0) = -\bar{P}_0\bar{D}^N(\bar{Q}, \bar{g}) + \bar{R}_2(\bar{Q}, \bar{Q}_0, \bar{g}, \bar{g}_0),$$

where we use the notation $\bar{Q} = (\bar{Q}, \bar{Q}_W)$ and $\bar{R}_2$ is second order term provided in Appendix D of [114]. Since $\bar{P}_0$ is defined as a mixture $1/N\sum_i P_{0,i}$, and combined with the fact that our TMLE solves the above efficient score equation, we obtain:

$$\bar{\Psi}_N(\bar{Q}_N^*) - \bar{\Psi}_N(\bar{Q}_0) = \frac{1}{N}\sum_i \left[\bar{D}^N(\bar{Q}_N^*, \bar{g}_N)(O_i^s) - P_{0,i}\bar{D}^N(\bar{Q}_N^*, \bar{g}_N)\right] + \bar{R}_{2,N}.$$

By having fast enough rates for $\bar{g}_N$ and $\bar{Q}_N^*$, one can show that $R_{2,N} = o_P(N^{-1/2})$ and by the empirical process and asymptotic equicontinuity analysis conducted in [63], using the same conditions as in Theorem 2 of [63], it follows that this empirical process applied to estimated $\bar{D}^N$ is up to $o_P(N^{-1/2})$ equal to the empirical process for the fixed limit, where we use $\bar{D}_0^N(O_i^s)$ to denote this limit and we have:

$$\bar{\Psi}_N(\bar{Q}_N^*) - \bar{\Psi}_N(\bar{Q}_0) = \frac{1}{N}\sum_i \left[\bar{D}_0^N(O_i^s) - P_{0,i}\bar{D}_0^N\right] + o_P(N^{-1/2}).$$

Finally, using the analogue analysis to the one conducted in [63], we can show that the above process converges to a normal limiting distribution at $\sqrt{N}$ rate, with its asymptotic variance given by the following limit:

$$\sigma^2_{0,\bar{g}^*} = \lim_{N\to\infty} \frac{1}{N}\sum_{i,j} R(i,j)E[\bar{D}_0^N(O_i^s)\bar{D}_0^N(O_j^s)],$$

for $(i,j) \in \{1,\ldots,N\}^2$ and $R(i,j) = 1$ when $F_i \cap F_j \neq \emptyset$, and $R(i,j) = 0$ otherwise, and we always have that $R(i,i) = 1$, for all $i = 1,\ldots,N$. We also refer to Theorem 2 in [63] for the full list of assumptions required for asymptotic normality of $\sqrt{N}(\bar{\Psi}_N(\bar{Q}_N^*) - \bar{\Psi}_N(\bar{Q}_0))$.

The above result provides us with inference for the parameter $\bar{\Psi}_N(\bar{Q}_0)$ (i.e., the data-adaptive parameter indexed by $\bar{g}_N^*$). We now perform the derivation which will also allow us to conduct inference for the parameter $\bar{\Psi}(\bar{Q}_0)$ defined with respect to true $\bar{g}_0^*$. Specifically, by

applying the same arguments as above, we can perform the following asymptotic expansion:

$$\begin{aligned}
&\bar{\Psi}_N(\bar{Q}_N^*) - \bar{\Psi}(\bar{Q}_0) \\
=&\bar{\Psi}_N(\bar{Q}_N^*) - \bar{\Psi}_N(\bar{Q}_0) + \bar{\Psi}_N(\bar{Q}_0) - \bar{\Psi}(\bar{Q}_0) \\
=&\frac{1}{N}\sum_i \left[\bar{D}_0^N(O_i^s) - P_{0,i}\bar{D}_0^N\right] + \frac{1}{N}\sum_i \left[\bar{D}_{0,i}^{\bar{g}^*}(O_i^s) - P_{0,i}\bar{D}_{0,i}^{\bar{g}^*}\right] + o_P(N^{-1/2}) \\
=&\frac{1}{N}\sum_i \left[\bar{D}_{i,0} - P_{0,i}\bar{D}_{i,0}\right] + o_P(N^{-1/2}),
\end{aligned}$$

where the contribution $D_0^{\bar{g}^*} = 1/N\sum_i \bar{D}_{0,i}^{\bar{g}^*}$ above was defined in the EIC in Theorem 1.3.3. We also note that the above expansion must hold for our TMLE $\bar{\Psi}(\bar{Q}_N^*, \bar{Q}_{W,N})$, since is solves the score equation given by:

$$\frac{1}{N}\sum_{i=1}^N \bar{D}_i^{\bar{g}^*}(\bar{Q}_N, \bar{g}_{NPMLE,N}^*)(\mathbf{W}) = 0.$$

By using the same set of arguments as before, we can now conclude that the above process will converge to a normal limiting distribution, i.e.,:

$$\sqrt{N}(\bar{\Psi}_N(\bar{Q}_N^*) - \bar{\Psi}(\bar{Q}_0)) \Rightarrow_d N(0, \sigma_0^2),$$

with σ_0^2 given by the following limit:

$$\sigma_0^2 = \lim_{N\to\infty} \frac{1}{N}\sum_{i,j} R(i,j) E[\bar{D}_0(O_i^s)\bar{D}_0(O_j^s)],$$

which includes additional terms $\bar{D}_{i,0}^{\bar{g}^*}$ and we assume $R(i,j) = 1$ when $F_i \cap F_j \neq \emptyset$, and $R(i,j) = 0$ otherwise, and we always have that $R(i,i) = 1$, for all $i = 1, \ldots, N$. We also note that this TMLE $\bar{\Psi}_N(\bar{Q}_N^*)$ will achieve the semi-parametric efficiency bound if both $\bar{Q}_N$ and $\bar{g}_N$ are consistent for $\bar{Q}_0$ and $\bar{g}_0$, under regularity conditions stated in [63], meaning that such an estimator is locally efficient at P_0^N.

Having a consistent estimator of σ_0^2 would yield asymptotically valid confidence intervals for ψ_0. A reasonable approach to estimating σ_0^2 is to plug in the estimates $\bar{Q}_N = (\bar{Q}_N, \bar{Q}_{W,N})$, $\bar{g}_N$, $\bar{g}_N^*$ and $\bar{\Psi}(\bar{Q}_N^*)$, obtaining the plug-in estimator $\bar{D}_N$ for $\bar{D}_0$, and then evaluate the above expectations $E[\bar{D}_N(O_i^s)\bar{D}_N(O_j^s)]$ with respect to their empirical counterparts, resulting in an estimator σ_N^2 of σ_0^2. We thus arrive at the following estimator of σ_0^2:

$$\sigma_N^2 \;=\; \frac{1}{N}\sum_{i,j} R(i,j)\left[\bar{D}_N(O_i^s)\bar{D}_N(\bar{O}_j)\right],$$

where

$$\begin{aligned}
\bar{D}_N(O_i^s) =&\bar{D}(\bar{Q}_N, \bar{g}_N, \bar{g}_N^*, \psi_N^*)(O_i^s) \\
=&\frac{\bar{g}_N^*}{\bar{g}_N}(A_i^s \mid W_i^s)\left[Y_i - \bar{Q}_N(A_i^s, W_i^s)\right] + \left[\Psi_i(\bar{Q}_N, \mathbf{Q}_{\mathbf{W},N}) - \bar{\Psi}_N(\bar{Q}_N^*)\right],
\end{aligned}$$

for

$$\Psi_i(\bar{Q}_N, \mathbf{Q}_{\mathbf{W},N}) = \int_{\mathbf{a}} \bar{Q}_N(a_i^s(\mathbf{a}, \mathbf{W}), w_i^s(\mathbf{W}))\mathbf{g}^*(\mathbf{a} \,|\, \mathbf{W})$$

and

$$\bar{\Psi}_N(\bar{Q}_N^*) = \frac{1}{N}\sum_{i=1}^{N} \int_{\mathbf{a}} \bar{Q}_N^*(a_i^s(\mathbf{a}, \mathbf{W}), w_i^s(\mathbf{W}))\mathbf{g}^*(\mathbf{a} \,|\, \mathbf{W}).$$

Given an estimator σ_N^2, one can construct a 95% confidence interval (CI) $\psi_N^* \pm 1.96\sigma_N/\sqrt{N}$, which, under the assumption of consistency of σ_N^2 for σ^2, will have correct asymptotic coverage. We note that the this estimator does not require assuming that $W_1, \ldots, W_N$ are independent, beyond the modeling assumptions on $\mathbf{Q}_{\mathbf{W},0} \in \mathcal{M}$ from Section 1.2. Furthermore, we know from the results in iid data that the such EIC-based confidence intervals will generally provide correct coverage when $\bar{Q}_N$ and $\bar{g}_N$ are correctly specified, and will be conservative if only $\bar{g}_N$ is specified correctly. Thus, we would expect our estimator σ_N^2 to be also conservative when the model for $\bar{Q}_0$ is misspecified, analogous to the result from the iid data, and we also test the validity of this conjecture with a simulation study.

1.6 Simulation study

We performed a network simulation study evaluating the finite sample bias and variance of the TMLE presented in Section 1.4. We also evaluated the finite sample coverage of the confidence intervals described in Section 1.5. In addition to TMLE, we also used the Inverse Probability Weighted (IPTW) estimator and the G-computation substitution estimator, where both of these estimators are defined below. All estimation was performed in **R** language [96] using a stand-alone package *tmlenet* [116]. The results are reported for networks consisting of $N = 500$ and $N = 1,000$ observations. The estimation was repeated by sampling $10,000$ datasets. Due to computing time limitations, each unit in the network was allowed to be connected to at most two other units (at most two friends in F_i, for each $i = 1, \ldots, N$). However, we note that since the same estimand would generally be obtained only once when using the actual observed data, one should be able to employ the *tmlenet* **R** package for estimation in more realistic network datasets where observed units may have much higher degrees of connectivity. The data generating distribution used in these simulations is described in more detail in Appendix 1.10. Briefly, we first sampled N iid baseline covariates, $\mathbf{W} = (W_1, \ldots, W_N)$. For each unit i, we then generated F_i by first sampling its size, $|F_i|$, uniformly from $\{0, 1, 2\}$, followed by uniform sampling without replacement of F_i from the population of $N-1$ units (all units, except i). The network-induced dependence among units was then simulated in the following manner. Each treatment A_i was sampled as a Bernoulli random variable, with its probability of success depending on the baseline covariate values of units in $F_i \cup \{i\}$. Similarly, each outcome Y_i was sampled as a Bernoulli random variable, with its probability of success depending on the baseline covariate values and treatments of units in $F_i \cup \{i\}$. The probability of success for each A_i was defined as a

logit-linear function of the 2-dimensional summary $(W_i, \sum_j W_j : j \in F_i)$, given as:

$$P_0(A_i = 1 \,|\, W_i^s) = \text{expit}(\alpha_0 + \alpha_1 W_i + \alpha_2 \sum_{j \in F_i} W_j).$$

Similarly, the probability of success for each Y_i was defined as a logit-linear function of the 4-dimensional summary $(W_i, \sum_j W_j, A_i, \sum_j A_j : j \in F_i)$, given as:

$$\bar{Q}_0(A_i^s, W_i^s) = \text{expit}(\beta_0 + \beta_1 A_i + \beta_2 \sum_{j \in F_i} A_j + \beta_1^{'} W_i + \beta_2^{'} \sum_{j \in F_i} W_j).$$

In contrast, the estimation of the common-in-i $\bar{Q}_0$ and the mixture density $\bar{g}_0$ was based on non-parametrically defined summary measures, i.e., we let $W_i^s = (|F_i|, W_i, W_j : j \in F_i)$ and $A_i^s = (A_i, A_j : j \in F_i)$, such that, for all i, $|W_i^s| = 4$ and $|A_i^s| = 3$. Whenever unit i had fewer than 2 friends ($|F_i| < 2$), the remainders of W_i^s and A_i^s were filled with zeros to ensure the same summary measure dimensionality across i. The common-in-i model $\bar{Q}_N$ for $\bar{Q}_0$ was then estimated by fitting a logistic regression model to a pooled sample of N units, using covariates (A_i^s, W_i^s). The estimation of the conditional mixture density $\bar{g}_0(a^s \,|\, w^s)$ proceeded as follows. First, for any $(a^s, w^s) \in (\bar{\mathcal{A}}^s \times \bar{\mathcal{W}}^s)$, such that $a^s \in \{0,1\}^3$ and $a^s = (a^s(1), a^s(2), a^s(3))$, we factorized $P(\bar{A}^s = a^s \,|\, \bar{W}^s = w^s)$ as:

$$\begin{aligned} P(\bar{A}^s = a^s \,|\, \bar{W}^s = w^s) &= P(\bar{A}^s(1) = a^s(1), \bar{A}^s(2) = a^s(2), \bar{A}^s(3) = a^s(3) \,|\, \bar{W}^s = w^s) \\ &= P(\bar{A}^s(1) = a^s(1) \,|\, \bar{W}^s = w^s) \\ &\quad \times P(\bar{A}^s(2) = a^s(2) \,|\, \bar{A}^s(1) = a^s(1), \bar{W}^s = w^s) \\ &\quad \times P(\bar{A}^s(3) = a^s(3) \,|\, \bar{A}^s(1) = a^s(1), \bar{A}^s(2) = a^s(2), \bar{W}^s = w^s). \end{aligned}$$

We then fit three separate logistic regression models, each estimating one of the factors in the above factorization, as if we were fitting common-in-i models using an iid sample of N observations (A_i^s, W_i^s). That is, the first factor $P(\bar{A}^s(1) = 1 | \bar{W}^s)$ was fit as if we were estimating a common-in-i model $P(A_i^s(1) = 1 | W_i^s)$ for N iid observations $(A_i, W_i^s)_{i=1}^N$ (note that $A_i^s(1) = A_i$). Similarly, the second factor was fit as if we were estimating a common-in-i model for $P(A_i^s(2) = 1 | A_i, W_i^s)$, and so on. The resulting three fits were then combined in order to obtain the estimate $\bar{g}_N(a^s | w^s)$ of $\bar{g}_0(a^s | w^s)$. We estimated $\bar{g}_0^*$ in a similar way, except that we first sampled a large dataset of observations (A_i^*, W_i) from $\mathbf{g}^*$ and $\mathbf{Q}_{\mathbf{W},N}$, for $i = 1, \ldots, mN$, then constructed the summary data $W_i^s = (W_i, W_j : j \in F_i)$, $A_i^{*s} = (A_i^*, A_j^* : j \in F_i)$, and finally estimated $\bar{g}_N^*$ by factorizing $P(\bar{A}^* = a^s \,|\, \bar{W}^s = w^s)$ into three factors and fitting three logistic regressions to a pooled sample (A_i^*, W_i) of mN observations.

The stochastic intervention $\mathbf{g}^*(\mathbf{A}|\mathbf{W})$ was defined as a common-in-i intervention g_p^* on each A_i, which assigned $A_i = 1$ with some constant probability p, i.e., $P(A_i^* = 1) = p$. Our target parameter was then defined as the sample-average of N outcomes under g_p^*, where we use $\psi_0(g_p^*)$ to denote the parameter's true value. In our simulations we then estimated a discrete dose response curve $\{\psi_0(g_p^*)\}$ for $p \in [0, 0.1, \ldots, 0.9, 1]$. We also truncated the

observation-specific weights $\bar{g}_N^*(a^s \mid w^s)/\bar{g}_N(a^s \mid w^s)$ when their values exceeded 10^5. Finally, the confidence intervals for the TMLE were constructed based on variance estimator σ_N^2 from Section 1.5. Lastly, we compared σ_N^2 with an alternative asymptotic variance estimator $\tilde{\sigma}_N^2$ presented in [63], which requires the consistency of $\bar{Q}_N$ and $\bar{g}_N$ and is given by:

$$\tilde{\sigma}_N^2 = \frac{1}{N}\left[\sum_{i=1}^{N} \frac{\bar{g}_N^*}{\bar{g}_N}(A_i^s, W_i^s)(Y_i - \bar{Q}_N(A_i^s, W_i^s))\right]^2 + \frac{1}{N}\sum_{i_1,i_2=1}^{N} f_{W,i_1} f_{W,i_2} I(F_{i_1} \cap F_{i_2} \neq \emptyset),$$

for

$$f_{W,i} = \int_a \bar{Q}_N(a_i^s(a), w_i^s(W)) g_i^*(a|W) - \int_w \left[\int_a \bar{Q}_N(a_i^s(a), w_i^s(W)) g_i^*(a|w)\right] Q_{W,N}(w).$$

IPTW (Horvitz-Thompson) estimator

The IPTW estimator is based on the TMLE weights $\bar{g}_0/\bar{g}_0^*$ from Section 1.4 and is defined as the weighted average of the observed outcomes Y_i, weighted by $\bar{g}_N^*/\bar{g}_N$:

$$\psi_{IPTW,N} = \frac{1}{N}\sum_{i=1}^{N} Y_i \left[\frac{\bar{g}_N^*}{\bar{g}_N}(A_i^s \mid W_i^s)\right],$$

where $\bar{g}_N$ is an estimator of the conditional mixture $\bar{g}_0(\mathbf{G}_0, \mathbf{Q}_{\mathbf{W},0})$ defined in Section 1.3 and $\bar{g}_N^*$ is an estimator of $\bar{g}_0^*(\mathbf{G}^*, \mathbf{Q}_{\mathbf{W},0})$, also defined in Section 1.3. The estimators $\bar{g}_N$ and $\bar{g}_N^*$ are described in general in Section 1.4 and are also described above for the case of non-parametrically defined summary measures. We also conducted inference for $\psi_{IPTW,N}$ by relying on the same ideas described in Section 1.5. That is, we used the iid-data influence curve $IC(\bar{P}_0)$ of $\psi_{IPTW,N}$ in a model that assumes $\bar{g}_0^*$ and $\bar{g}_0$ are known, characterizing the asymptotic variance of $\psi_{IPTW,N}$ by the following limit:

$$\sigma_{IPTW,0}^2 = \lim_{N\to\infty} \frac{1}{N}\sum_{i,j} R(i,j) E_{P_0}[IC(\bar{P}_0)(O_i) IC(\bar{P}_0)(O_j)],$$

with $R(i,j) = 1$ when $F_i \cap F_j \neq \emptyset$, and $R(i,j) = 0$ otherwise. Replacing the unknown components of $\bar{P}_0$ in $IC(\bar{P}_0)$ with corresponding estimates, we then obtained the following estimator $\sigma_{IPTW,N}^2$ of $\sigma_{IPTW,0}^2$:

$$\sigma_{IPTW,N}^2 \quad = \quad \frac{1}{N}\sum_{i,j}\left[R(i,j)(Y_i \frac{\bar{g}_N^*}{\bar{g}_N}(A_i^s \mid W_i^s) - \psi_{IPTW,N})(Y_j \frac{\bar{g}_N^*}{\bar{g}_N}(A_j^s \mid W_j^s) - \psi_{IPTW,N})\right],$$

for $IC(\bar{P}_0)(O_i^s) = Y_i\,[\bar{g}_0^*/\bar{g}_0]\,(A_i^s \mid W_i^s) - \psi_0$. We then constructed 95% CIs as $\psi_{IPTW,N} \pm 1.96\sigma_{IPTW,N}^2/\sqrt{N}$.

G-computation estimator

The G-computation substitution estimator $\psi_{GCOMP,N} = \Psi(\bar{Q}_N, \mathbf{Q}_{\mathbf{W},N})$ for ψ_0 is based on the un-targeted model $\bar{Q}_N$ for the common-in-i conditional expectation of Y_i, as a function of the summary data (A_i^s, W_i^s). Given stochastic intervention $\mathbf{g}^*$, the G-computation estimator is obtained as:

$$\psi_{GCOMP,N} = \frac{1}{N}\sum_{i=1}^{N}\int_{\mathbf{a}} \bar{Q}_N(a_i^s(\mathbf{a},\mathbf{W}), W_i^s)\mathbf{g}^*(\mathbf{a}\,|\,\mathbf{W})$$

where $\mathbf{Q}_{\mathbf{W},N}$ is a NPMLE that puts mass 1 on observed vector $\mathbf{W}$. Evaluation of this estimator is equivalent to the Monte Carlo integration procedure described for the TMLE ψ_N^* in Section 1.4, except that we use the initial estimator $\bar{Q}_N$ for $\bar{Q}_0$ instead of its targeted version $\bar{Q}_N^*$. The asymptotic variance of $\psi_{GCOMP,N}$ was not estimated and no CIs were constructed.

Results

Our simulations compared three different model specification scenarios for $\bar{Q}_0$ and $\bar{g}_0^*/\bar{g}_0$: "*(a) Q and $\bar{g}^*/\bar{g}$ correct*" indicates that the models for both estimators, $\bar{Q}_0$ and $\bar{g}_0^*/\bar{g}_0$, were correctly specified; "*(b) only $\bar{g}^*/\bar{g}$ correct*" indicates that the model for the estimator of $\bar{Q}_0$ was misspecified, while the model for the estimator of $\bar{g}_0^*/\bar{g}_0$ was specified correctly; finally, "*(c) only Q correct*" indicates that the model for the estimator of $\bar{Q}_0$ was specified correctly, while the model for the estimator of $\bar{g}_0^*/\bar{g}_0$ was misspecified. Figures 1.1 and 1.2 present the simulation results for finite sample bias and empirical variance. Bias was plotted as the estimate minus the true parameter value $(\psi_N(g_p^*) - \psi_0(g_p^*))$, with different stochastic interventions g_p^* presented on the x-axis as "% Treated". Overall, our simulation results suggest that TMLE performs well in finite samples with dependent observations. We were able to demonstrate the double robustness property of TMLE, with it being unbiased in each of the three considered scenarios. Our results also indicate that the other two estimators are unbiased for scenario (a), but can perform poorly in alternative scenarios (b) and (c). Overall, we found the IPTW estimator to be the most variable and also most susceptible to near-positivity violations.

The coverage results are presented in Figures 1.3-1.5, where we plotted the 95% CI coverage for various asymptotic variance estimators, along with the mean CI length. We first compared the TMLE coverage of our proposed variance estimator, σ_N^2, from Section 1.5 to the TMLE coverage based on the iid variance estimate $\sigma_{IID,N}^2$ that made no adjustments for correlated observations, i.e., $\sigma_{IID,N}^2$ is the EIC-based variance estimator that assumes data are iid. Our results in Figure 1.3 indicate that $\sigma_{IID,N}^2$ tended to under-estimate the variance of TMLE, resulting in CIs that were too narrow for both sample sizes. We expect the coverage issues for $\sigma_{IID,N}^2$ to become even more pronounced when the between-unit dependence increases, as may be the case in more realistic network scenarios with units having much higher degrees of connectivity.

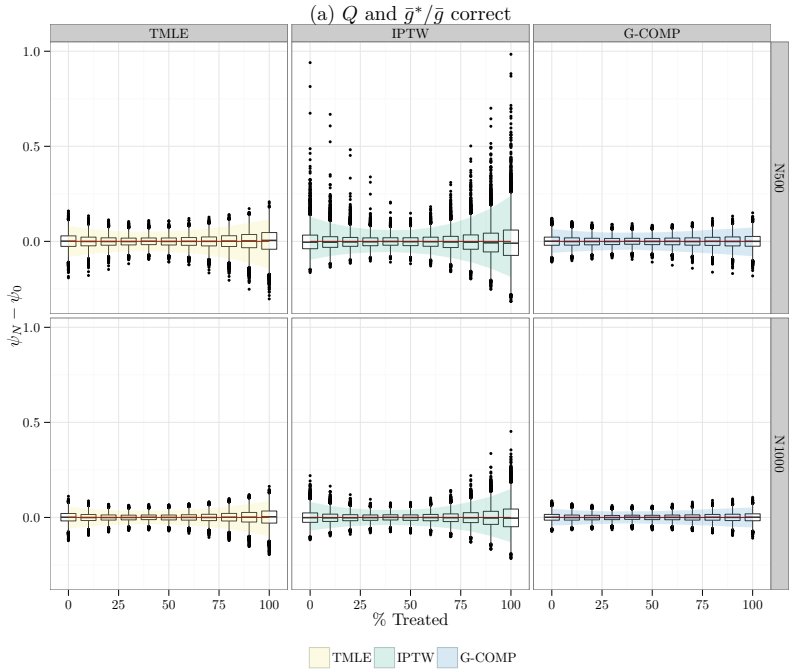

Figure 1.1: Empirical distributions for TMLE, IPTW and G-COMP, centered at the truth and estimated over 10,000 simulated data sets of size 500 (top row) and size 1,000 (bottom row) for scenario (a) - correctly specified Q and $\bar{g}^*/\bar{g}$. Colored ribbons mark the 2.5th to 97.5th percentile ranges of the estimands. The centered IPTW estimates outside the range of ± 1 were removed.

In addition, the CIs for our dependent-data variance estimate σ_N^2 become conservative when $\bar{Q}_N$ was misspecified. The latter result was expected based on the predictions from the semi-parametric efficiency theory for iid data. In Figure 1.4 we compared the coverage of IPTW with that of TMLE. Finally, we compared the TMLE coverage for our dependent-data variance estimate σ_N^2 to the alternative asymptotic variance estimate $\tilde{\sigma}_N^2$ from [63]. The simulation results of this comparison in Figure 1.5 show nearly identical coverage under correctly specified $\bar{Q}_N$. However, when $\bar{Q}_N$ is misspecified, the two estimators behaved differently, with $\tilde{\sigma}_N^2$ showing slightly lower coverage for some sections of the estimated dose response curve. We also note that near positivity violations will generally increase the variability of our estimators. In particular, one would expect the near positivity violations to be more pronounced closer to the tail-ends of the discrete dose response curve $\{\psi_0(g_p^*)\}$, namely, for values of p close to 0 or 1. Indeed, this is also demonstrated in our simulations, where we noted increasing variability of all estimators closer to the edges of the estimated dose response curve, which also contributes to a small drop in coverage.

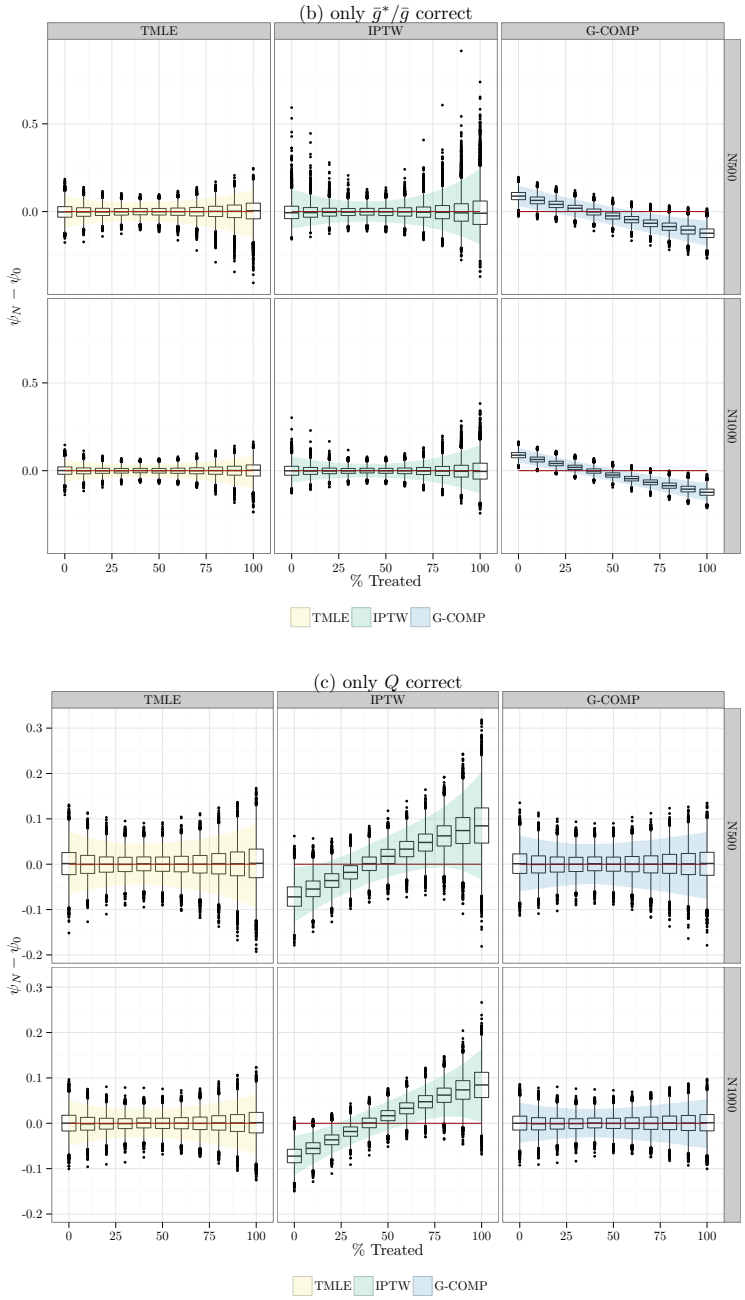

Figure 1.2: Empirical distributions for TMLE, IPTW and G-COMP, centered at the truth and estimated over 10,000 simulated data sets of size 500 (top row) and size 1,000 (bottom row) for scenarios: (b) - only $\bar{g}^*/\bar{g}$ correctly specified; and (c) - only Q correctly specified. Colored ribbons mark the 2.5th to 97.5th percentile ranges of the estimands. The centered IPTW estimates outside the range of ± 1 were removed.

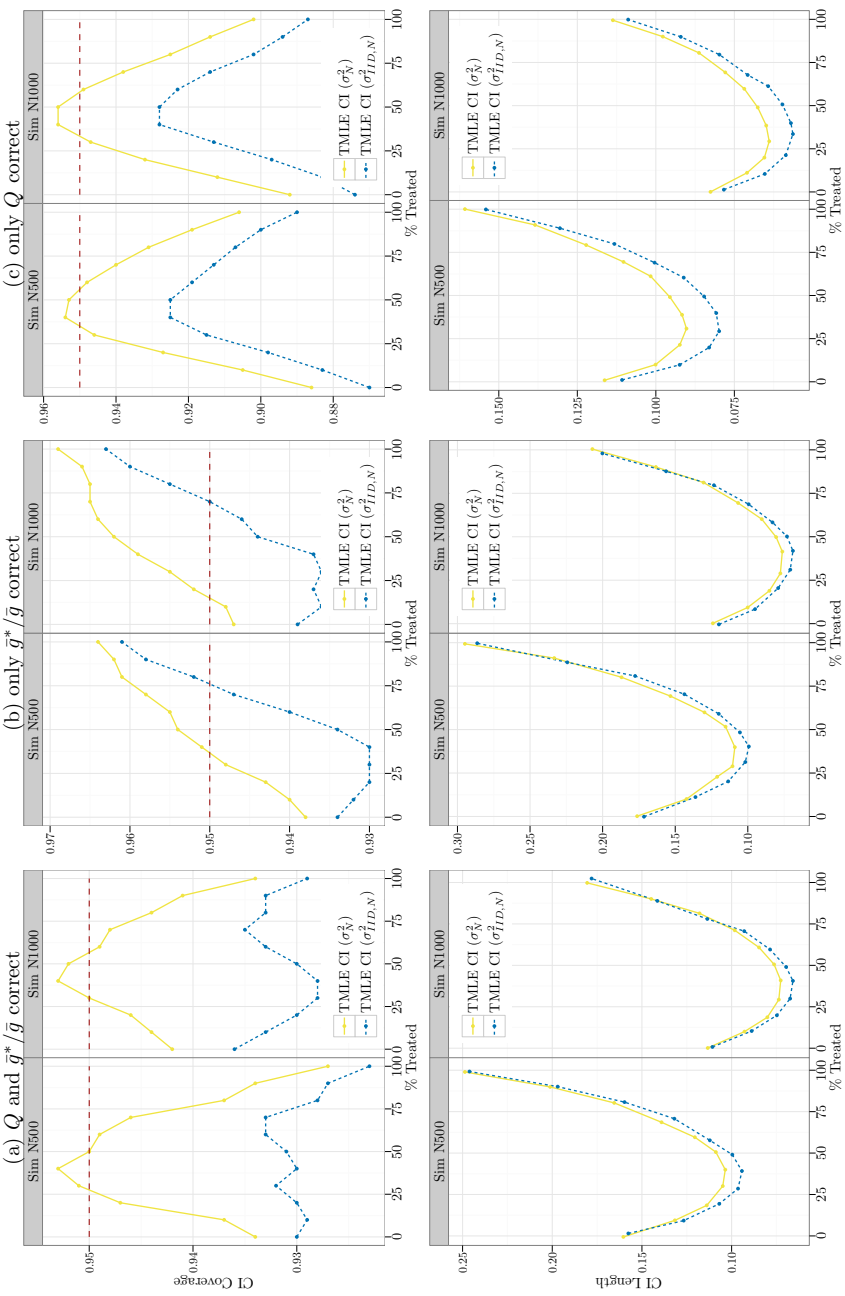

Figure 1.3: Comparing the 95% CI coverage (top row) and length (bottom row) for the TMLE using two alternative variance estimates: σ_N^2 - variance estimate that correctly adjusts for the dependence between observations; $\sigma_{IID,N}^2$ - iid variance estimate that ignores the dependence between observations. The estimates are obtained from 10,000 simulated data sets of size 500 ('Sim N500') and size 1,000 ('Sim N1000'). Scenarios: (a) - correctly specified Q and $\bar{g}^*/\bar{g}$; (b) - only $\bar{g}^*/\bar{g}$ correctly specified; and (c) - only Q correctly specified.

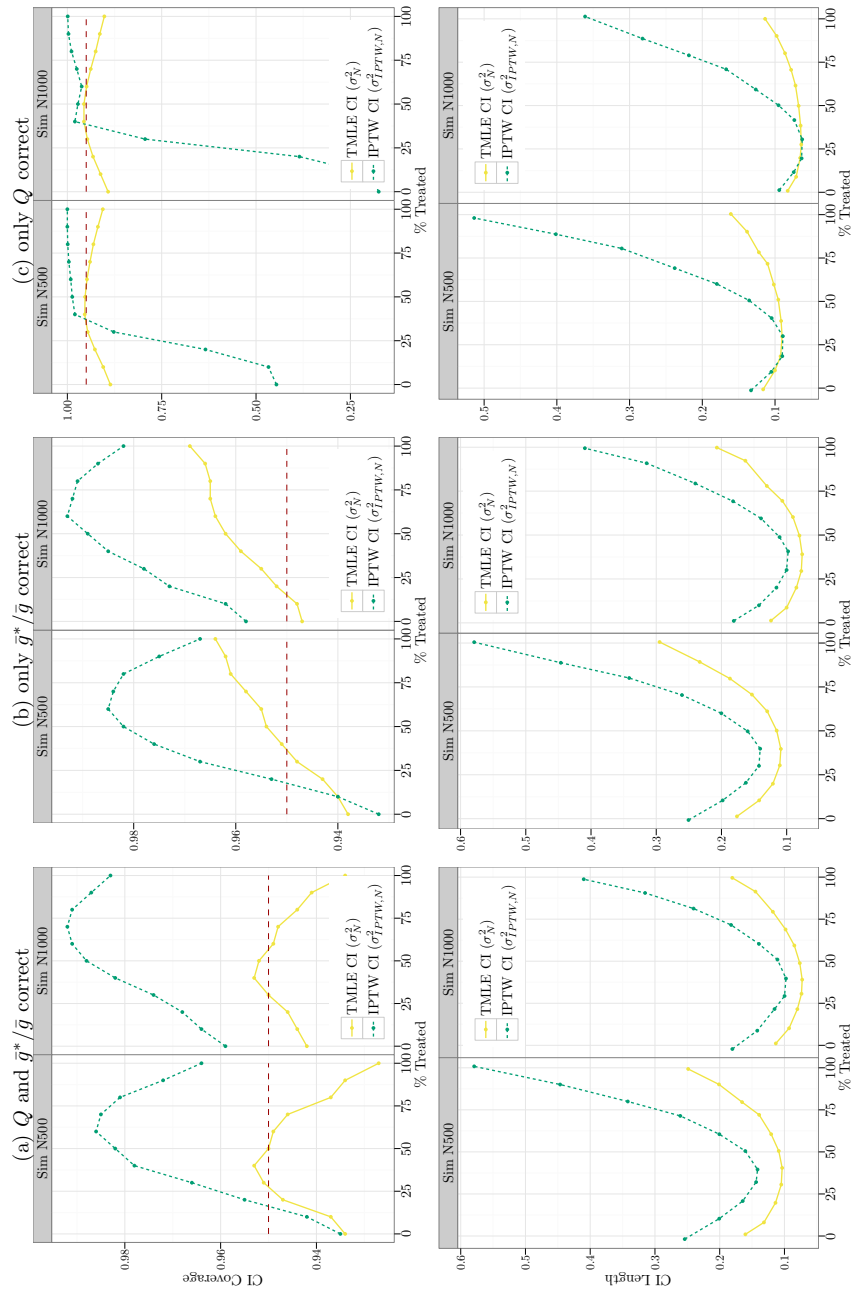

Figure 1.4: Comparing the 95% CI coverage (top row) and length (bottom row) for TMLE (variance estimate σ^2_N) and IPTW (variance estimate $\sigma^2_{IPTW,N}$). The estimates are obtained from 10,000 simulated data sets of size 500 ('Sim N500') and size 1,000 ('Sim N1000'). Scenarios: (a) - correctly specified Q and $\bar{g}^*/\bar{g}$; (b) - only $\bar{g}^*/\bar{g}$ correctly specified; and (c) - only Q correctly specified.

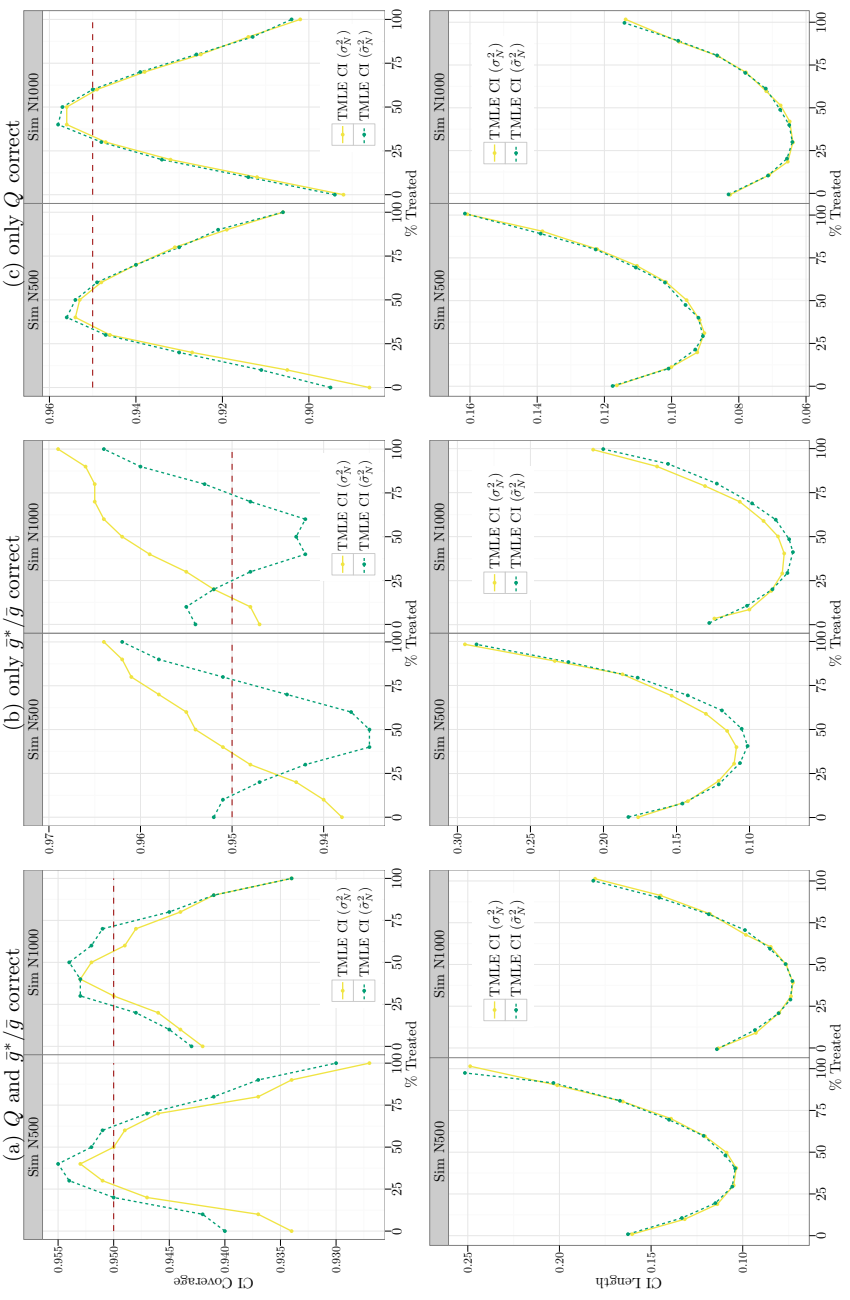

Figure 1.5: Comparing the 95% CI coverage (top row) and length (bottom row) of the two TMLE variance estimates, σ_N^2 and $\tilde{\sigma}_N^2$, both of which adjust for the dependence between units, but the latter variance estimate assumes correctly specified $\bar{Q}_N$. The estimates are obtained from 10,000 simulated data sets of size 500 ('Sim N500') and size 1,000 ('Sim N1000'). Scenarios: (a) - correctly specified Q and $\bar{g}^*/\bar{g}$; (b) - only $\bar{g}^*/\bar{g}$ correctly specified; and (c) - only Q correctly specified.

1.7 Intervening on groups of friends and intervening on network

Estimation for an arbitrary collection of stochastic interventions

We now show that the estimation framework presented thus far can be easily adapted to the estimation of the sample-average treatment effects for an arbitrary collection of i-specific stochastic interventions $g^*_{F_i}$, where each $g^*_{F_i}$ may intervene on the treatments of i's friends in addition to intervening on the treatment of i itself. A collection of such interventions involving all units, $\{g^*_{F_i} : i = 1, \ldots, N\}$, generally cannot be described by a single intervention $\mathbf{g}^*$ on $\mathbf{A} = (A_1, \ldots, A_N)$ given $\mathbf{W} = (W_1, \ldots, W_N)$. For example, consider the problem of estimating the direct average treatment effect in a network of N connected individuals, where we define each $g^*_{F_i}$ by intervening on a unit-specific treatment, A_i, setting it to a constant (zero or one), but leave unchanged the distribution of A_j for i's friends $j \in F_i$ intact. That is, we assume the intervention for each A_j is stochastically sampled from its observed distribution $G_0(A_j|W^s_j)$ or instead deterministically set to its observed value a_j. This type of direct effect parameter has been previously explored in spillover studies, for example, in the study of the effects of deworming among rural Kenyan primary schools by [72] and in its replication study by [30]. We are interested in estimation of the following target parameter,

$$\psi_0 = \Psi(P^N_0) = \frac{1}{N} \sum_{i=1}^{N} E_{\mathbf{q}_0, g^*_{F_i}} \left[Y_i^{g^*_{F_i}} \right],$$

defined as the average of expectations of $Y_i^{g^*_{F_i}}$, where each outcome $Y_i^{g^*_{F_i}}$ is generated under the i-specific post-intervention distribution that replaces the observed treatment allocation for i and $j \in F_i$ with $g^*_{F_i}$ as just described. Clearly the collection of such i-specific interventions across all N units is incompatible with a single joint stochastic intervention $\mathbf{g}^*$ on $\mathbf{A}$ given $\mathbf{W}$, since the intervention $g^*_{F_j}$ for $j \neq i$ and $j \in F_i$ requires setting A_j to a constant one or zero, while the intervention $g^*_{F_i}$ requires that A_j is randomly sampled from g_0 or is set to a_j. Nonetheless, this target parameter ψ_0 remains well-defined with respect to a collection $\{g^*_{F_i} : i = 1, \ldots, N\}$, and we may apply the same arguments as in Section 1.3, noting that ψ_0 can be equivalently written as:

$$\begin{aligned}
\psi_0 &= \frac{1}{N} \sum_{i=1}^{N} \int_{\mathbf{a},\mathbf{w}} \bar{Q}_0(a^s_i(\mathbf{a},\mathbf{w}), w^s_i(\mathbf{w})) g^*_{F_i}(\mathbf{a}|\mathbf{w}) \mathbf{q}_{\mathbf{W},0}(\mathbf{w}) d\mu(\mathbf{a},\mathbf{w}) \\
&= \frac{1}{N} \sum_{i=1}^{N} \int_{a^s,w^s} \bar{Q}_0(a^s, w^s) h^*_{i,0}(a^s, w^s) d\mu(a^s, w^s) \\
&= \int_{a^s,w^s} \bar{Q}_0(a^s, w^s) \bar{h}^*_0(a^s, w^s) d\mu(a^s, w^s),
\end{aligned}$$

where $h^*_{i,0}(g^*_{F_i}, \mathbf{q}_{\mathbf{W},0})$ is the density determined by $g^*_{F_i}(\mathbf{a}|\mathbf{w})$, $\mathbf{q}_{\mathbf{W},0}$ and the i-specific summary measures $a^s_i(\mathbf{a},\mathbf{w}), w^s_i(\mathbf{w})$, and $\bar{h}^*_0$ is a mixture of $h^*_{i,0}$, defined as $\bar{h}^*_0(a^s, w^s) := 1/N \sum_{i=1}^{N} h^*_{i,0}(a^s, w^s)$.

We also note that when $(\mathbf{W}, \mathbf{A})$ are discrete, one obtains:

$$h^*_{i,0}(a^s, w^s) = \int_{\mathbf{a},\mathbf{w}} I(a^s_i(\mathbf{a}, \mathbf{w}) = a^s, w^s_i(\mathbf{w}) = w^s) g^*_{F_i}(\mathbf{a}|\mathbf{w}) \mathbf{q}_{\mathbf{W},0}(\mathbf{w}) d\mu_{a,w}(\mathbf{a}, \mathbf{w}).$$

Thus, this new target parameter $\Psi(P^N_0)$ can still be represented by an equivalent mixture mapping $\bar{\Psi}(\bar{P}_0)$ from Theorem 1.3.1 and hence, the efficient influence curve of this new $\Psi(P^N_0)$ is given by the same $\bar{D}$ from Theorem 1.3.3. In the above, we also assumed that such i-specific densities $h^*_{i,0}(a^s, w^s)$ are well-defined with respect to some common dominating measure $\mu_{a,w}$, with $\bar{h}^*_0$ being factorized as $\bar{h}^*_0(a^s, w^s) = \bar{g}^*_0(a^s|w^s)\bar{q}_{W,0}(w^s)$, which provides the definition of $\bar{g}^*_0$. Hence, the TMLE framework from Section 1.4 can be directly applied to estimation of these types of parameters, with the only modification that we now require that each i-specific summary A^{*s}_i is sampled conditionally from its i-specific intervention $g^*_{F_i}$ on $\mathbf{A}$ given $\mathbf{W}$. In particular, we use Lemma 1.4.1 to obtain a reasonable approximation for $\bar{g}^*_0$ by modifying it's direct estimator from Section 1.4 in the following manner: First, obtain a large simulated dataset of i-specific summaries (A^{*s}_i, W^s_i), for $i = 1, \ldots, N$, where each A^{*s}_i is derived by sampling $(\mathbf{A}^{g^*_{F_i}}, \mathbf{W})$ from $(g^*_{F_i}, \mathbf{Q}_{\mathbf{W},N})$ and then applying the summary measure mapping $a^s_i(\mathbf{A}^{g^*_{F_i}}, \mathbf{W})$. Next, fit an estimator $\bar{g}^*_N$ of $\bar{g}^*_0$ by treating the simulated sample (A^{*s}_i, W^s_i) as iid, applying the same maximum likelihood-based approach as earlier. Similarly, the Monte Carlo evaluation step for the TMLE ψ^*_N from Section 1.4 is modified to take into account these i-specific interventions $g^*_{F_i}$. That is, instead of sampling $(\mathbf{A}^*, \mathbf{W})$ from $\mathbf{g}^*$, each Monte Carlo iteration j now consists of sampling $\mathbf{A}^{g^*_{F_i}}_j = (A^{g^*_{F_i}}_{j,1}, \ldots, A^{g^*_{F_i}}_{j,N})$ from $g^*_{F_i}(\mathbf{a}|\mathbf{W})$, for each $i = 1, \ldots, N$, conditional on the observed $\mathbf{W}$, with a resulting dataset of N summary observations $(A^{*s}_{j,i}, W^s_i)$ constructed by applying the i-specific mappings $A^{*s}_{j,i} := a^s_i(\mathbf{A}^{g^*_{F_i}}_j, \mathbf{W})$, for $i = 1, \ldots, N$.

Estimation for interventions on the network structure F

Overview. Observed network data structure. We note that our framework can be also applied to estimation of the effect of intervening on the network structure. Suppose that we observe at baseline some initial network $\mathbf{F}^0 = (F^0_1, \ldots, F^0_N)$ for the community of N connected units and we are treating $\mathbf{F}^0$ as fixed. As before, we are assuming that the maximum number of friends for each unit (i.e., the dimensionality of each F^0_i) is bounded by some constant K that doesn't depend on N. We then collect data on N baseline covariates $\mathbf{W} = (W_1, \ldots, W_N)$, followed by a random draw of another network profile $\mathbf{F} = (F_1, \ldots, F_N)$ and the outcomes $\mathbf{Y} = (Y_1, \ldots, Y_N)$, where each F_i is now based on the initial network F^0_i. Thus, the observed data on N connected units is given by $\mathbf{O} = (\mathbf{F}^0, \mathbf{W}, \mathbf{F}, \mathbf{Y})$ and we assume the exposure for each unit i is given by the set F_i. Since we are interested in interventions which will modify the observed network profile $\mathbf{F}$ (e.g., adding or removing some friends in each F_i) it is natural to allow F_i to be random, but driven by i's own covariates and the covariates of i's friends from F^0_i. Thus, we assume that the i-specific conditional distribution

$G_{F_i,0}$ for F_i given $(\mathbf{F}^0, \mathbf{W})$ only depends on the initial network offset F_i^0 and the baseline covariates $(W_i, W_j : j \in F_i^0)$. Furthermore, we assume its conditional density $g_{F_i,0}(F_i \,|\, \mathbf{F}^0, \mathbf{W})$ is well-defined. We also assume that $Q_{Y,0}$, the common-in-i conditional distribution of Y_i given $\mathbf{W}$, depends only on $(W_i, W_j : j \in F_i)$, i.e., units j from this newly drawn friend set F_i.

Network interventions and target parameter. We follow the framework outlined in Section 1.2 and define the intervention on a network profile $\mathbf{F}$ as a user-supplied density $\mathbf{g}^*(\mathbf{F}^* \,|\, \mathbf{W})$ that replaces the observed conditional density $\mathbf{g}_0(\mathbf{F} \,|\, \mathbf{W})$, where we also assumed that the initial network offset $\mathbf{F}^0$ is included in $\mathbf{W}$. Alternatively, we could also follow Section 1.7 and define our intervention as a collection of the user-supplied i-specific densities $\{g^*_{F_i} : i = 1, \ldots, N\}$, where each $g^*_{F_i}(F_i^* \,|\, \mathbf{W})$ replaces the true i-specific density $g_{F_i,0}(F_i \,|\, \mathbf{W})$. As noted in Section 1.7, a collection of such i-specific stochastic interventions generally cannot be described by a single multivariate intervention $\mathbf{g}^*$ on $\mathbf{F} = (F_1, \ldots, F_N)$ given $\mathbf{W} = (W_1, \ldots, W_N)$ and may result in an incompatible network intervention. Nonetheless, these are still well-defined interventions and we note that the target parameter indexed by such i-specific interventions $g^*_{F_i}(F_i^* \,|\, \mathbf{W})$ is still well-defined. We also note that the types of interventions we will consider will generally use the current network sets F_i as inputs, to produce intervened network sets F_i^*. Therefore, we are concerned here with stochastic interventions which depend on the current sets F_i. Even if the intervention itself is a deterministic function of F_i (e.g., always remove the first friend), it is still stochastic by the nature of its dependence on F_i.

As a motivating example, consider an intervention defined for $i = 1, \ldots, N$ that removes certain friends $j \in F_i$ of each unit i when $\delta(W_j) \geq r$, where r is some pre-defined cutoff value and $\delta(W_j)$ is some user-defined function mapping W_j in $\mathbb{R}$ (e.g., $\delta(W_j)$ characterizes the baseline "risk-profile" of j). This intervention defines the post-intervention distribution that replaces $Q_{Y,0}(W_j : j \in F_i)$, i.e., the observed conditional distribution of Y_i given $\mathbf{W}$, with a new distribution $Q_{Y,0}(W_j : j \in F_i^*)$, where F_i^* is the intervened friend set such that the unit $k \in F_i^*$ only if $k \in F_i$ and $\delta(W_k) < r$. We now define our statistical parameter as the sample-average of the expected outcomes under the i-specific post-intervention distributions that replace each observed network allocations $g_{F_i,0}$ with such intervention densities $g^*_{F_i}$:

$$\Psi(P_0^N) = \frac{1}{N} \sum_{i=1}^{N} E_{\mathbf{q}_0, g^*_{F_i}} \left[Y_i^{g^*_{F_i}} \right].$$

Statistical model and dimension reduction assumptions. We have described how the interventions on the networks sets F_i fit within our previously outlined framework in Section 1.2, where F_i defines the exposure for each unit i. Our next step is concerned with dimensionality reduction, where we define the appropriate summary measures (W_i^s, A_i^s), for $i = 1, \ldots, N$, and then model conditional distribution of Y_i as a fixed-dimensional function of (W_i, A_i^s). In order to be able to intervene on F_i we have to assume that the conditional distribution $Q_{Y,0}$ depends on all $\mathbf{W}$ through an N dimensional set $(W_j I(j \in F_i) : j = 1, \ldots, N)$.

That is, we have so far assumed that $Q_{Y,0}$ is a function of $(W_i, W_j : j \in F_i)$. Replacing F_i with the intervened friend set F_i^* implies that $Q_{Y,0}$ becomes a function of $(W_i, W_j : j \in F_i^*)$. In that sense, Y_i depends on all $\mathbf{W}$, except that for most j, we have $I(j \in F_i) = 0$ and hence W_j makes no real contribution to $Q_{Y,0}$, unless $j \in F_i$. In summary, we change the dependence of Y_i on particular W_j in $\mathbf{W}$ by changing the composition of the set F_i and we only intervene on the way Y_i may depend on a particular W_j through indicators $I(j \in F_i)$, for $j = 1, \ldots, N$. This also implies that the conditional distribution of the outcome Y_i is now truly a function of the entire N dimensional set $\mathbf{W} = (W_1, \ldots, W_N)$, making estimation of $Q_{Y,0}$ particularly challenging. Thus, we first need to make additional simplifying assumptions which would allow us to estimate $Q_{Y,0}$.

For convenience, we now assume that $i \in F_i^0$, $i \in F_i$ and $i \in F_i^*$, for all $i = 1, \ldots, N$ (i.e., i is always connected to itself). Assume that the i's network draw F_i is always a finite dimensional augmentation of the initial network offset F_i^0. That is, the set of possible realizations of F_i is restricted to be within some close proximity of F_i^0. We note that the network profile $\mathbf{F}$ can be viewed as an $N \times N$ adjacency matrix of indicators and our assumptions imply that rather then allowing $\mathbf{F}$ to be any possible realization of an $N \times N$ adjacency matrix, we restrict $\mathbf{F}$ to finite-dimensional perturbations of matrix $\mathbf{F}^0$, allowing $\mathbf{F}$ to only change locally as a function of $\mathbf{F}_0$ and $\mathbf{W}$. For example, rather than allowing F_i to draw any new friend $j \in \{1, \ldots, N\} \backslash i$, one may assume that F_i is restricted to drawing a new friend j only when j is in a set $F_i^{0+} := \cup_{j \in \{F_i \cup i\}} F_j^0$. In this case we are assuming that in the new network realization F_i can only add friend j if i and j had at least one friend in common at baseline (i.e., there was at most 2nd degree of connectivity between i and j). Such an assumption implies that F_i is no longer of dimension N, but is rather of a fixed dimension that only depends on K. The set of possible network interventions on F_i, namely, each intervened F_i^*, is now similarly restricted to realizations of the same finite-dimensional set F_i^{0+}.

Having defined F_i as a realization of the finite dimensional set F_i^+, we define the unit's exposure A_i, for $i = 1, \ldots, N$, as a finite-dimensional set of indicators $(I(j \in F_i) : j \in F_i^{0+})$, namely, each A_i is a binary vector of the common-in-i dimension K^+, with each non-zero entry in A_i denotes which units in F_i^{0+} are actual friends of i. The i-specific network intervention A_i^* can be defined by directly intervening on the indicators $I(j \in F_i)$ in A_i. We define the baseline summary measure $W_i^s := w_i^s(\mathbf{W})$ as a finite dimensional set $(W_j : j \in F_i^{0+})$, for $i = 1, \ldots, N$. Additionally, we define the exposure summary measure $a_i^s(A_i, W_i^s)$ that depends on (A_i, W_i^s) only as a function of the set $(W_j I(j \in F_i) : j \in F_i^{0+})$, i.e.,

$$A_i^s := a_i^s(A_i, W_i^s) = a_i^s(W_j I(j \in F_i) : j \in F_i^{0+}),$$

for $i = 1, \ldots, N$, and we assume each $a_i^s(\cdot)$ maps into $\mathbb{R}^d$, for a common-in-i dimension d. Next, we model the conditional probability of the outcome Y_i as a common-in-i function $Q_{Y,0}$ that depends on $(\mathbf{A}, \mathbf{W})$ only as a function of the summary A_i^s, for $i = 1, \ldots, N$. Note that the crucial assumption that $a_i^s(\cdot)$ is a function of $(W_j I(j \in F_i) : j \in F_i^{0+})$ allows us to take full advantage of the dimensionality reduction due to $a_i^s(\cdot)$ and still define the target

parameters which actually correspond to the effects of intervening on the observed network realization $\mathbf{F}$. We may also assume that Y_i depends on some common-in-i summary $\tilde{w}^s(W_j : j \in F_i^{0+}) = \tilde{w}^s(W_i^s)$ that is unrelated to the new network draw F_i, and hence would not be affected by an intervention F_i^* on F_i. Note that such modeling restrictions on F_i now make it possible to estimate our parameter $\Psi(P_0^N)$.

With a slight abuse of notation, we use $g_{F_i,0}(A_i|\mathbf{W})$ to denote the conditional density of A_i given $\mathbf{W}$, and similarly, we use $g^*_{F_i}(A_i^* \,|\, \mathbf{W})$ to denote the i-specific stochastic intervention on A_i, implied by F_i^*. We also note that due to our modeling assumptions for the conditional distribution of F_i given $(\mathbf{F}^0, \mathbf{W})$, we have that $g_{F_i,0}(A_i|\mathbf{W}) = g_{F_i,0}(A_i|W_i^s)$ and $g^*_{F_i}(A_i|\mathbf{W}) = g^*_{F_i}(A_i|W_i^s)$, which also leads to the following representation of our target parameter:

$$
\begin{aligned}
\Psi(P_0^N) &= \frac{1}{N}\sum_{i=1}^{N} E_{Q_{W_i^s,0}}\left[E_{g^*_{F_i}}\left[\bar{Q}_0(a_i^s(A_i^*, W_i^s), \tilde{w}^s(W_i^s)) \,|\, W_i^s\right]\right] \\
&= \frac{1}{N}\sum_{i=1}^{N} \int_{a_i, w_i^s} \bar{Q}_0(a_i^s(a_i, w_i^s), \tilde{w}^s(w_i^s)) g^*_{F_i}(a_i|w_i^s) q_{W_i^s,0}(w_i^s) d\mu(a_i, w_i^s) \\
&= \frac{1}{N}\sum_{i=1}^{N} \int_{a^s, w^s} \bar{Q}_0(a^s, \tilde{w}^s(w^s)) h^*_{i,0}(a^s, w^s) d\mu(a, w^s) \\
&= \int_{a, w^s} \bar{Q}_0(a^s, \tilde{w}^s(w^s)) \bar{h}^*_0(a^s, w^s) d\mu(a^s, w^s) \\
&= \int_{a, w^s} \bar{Q}_0(a^s, \tilde{w}^s(w^s)) \bar{g}^*_0(a^s|w^s) \bar{q}_{W,0}(w^s) d\mu(a^s, w^s) \\
&= E_{\bar{q}_{W,0}}\left[E_{\bar{g}^*_0}\left[\bar{Q}_0(\bar{A}^s, \tilde{w}^s(\bar{W}^s)) \,|\, \bar{W}^s\right]\right].
\end{aligned}
$$

This shows that the new target parameter $\Psi(P_0^N)$ can be represented by an equivalent mixture mapping $\bar{\Psi}(\bar{P}_0)$ from Theorem 1.3.1. Consequently, the efficient influence curve of this new $\Psi(P_0^N)$ is given by the same $\bar{D}$ from Theorem 1.3.3. It follows that the estimation procedure described in Section 1.4 remains unchanged when estimating this new parameter $\Psi(P_0^N)$. As before, we also assume that the i-specific densities $\bar{h}^*_0(a^s, w^s)$ are well-defined with respect to some common dominating measure, and that $\bar{h}^*_0$ can be factorized as $\bar{h}^*_0(a^s, w^s) = \bar{g}^*_0(a^s|w^s)\bar{q}_W(w^s)$.

As we describe in an example below, given a particular context, one might assume that the summary measures A_i^s are of lower dimensionality than the identity mapping $(W_j I(j \in F_i) : j \in F_i^{0+})$. For instance, one may be able to define some low-dimensional summaries of $(I(j \in F_i)W_j : j \in F_i^{0+})$, which incorporate various features of unit i's network and the covariates of i's friends. One then has to assume that the conditional outcome model $Q_{Y,0}$ for each Y_i depends only on such low-dimensional features. Furthermore, if $a_i^Y(\cdot)$ and $\tilde{w}^s(\cdot)$ depend only on some subset of $(W_j : j \in F_i^{0+})$, then W_i^s can be also redefined as a summary of lower dimension that only includes the subset of $(W_j : j \in F_i^{0+})$ or the specific features of this subset. The direct estimation of the conditional mixtures $\bar{g}_0$ and $\bar{g}^*_0$ is then performed conditional on this lower-dimensional summary W_i^s.

Example. Suppose that we gather some initial data on a network of sexual partners ($\mathbf{F}^0 = F_1^0, \ldots, F_N^0$) in a community-based observational study of HIV risk factors. For each unit i, we measure the baseline covariates, W_i, which may include baseline HIV infection status (H_i) along with various risk factors. We also assume that after some period of time the network of sexual partners on each unit was measured again. This new network realization for unit i is denoted as set F_i and it defines our exposure of interest. Suppose that several months later data was collected on the binary outcome Y_i which indicated whether subject i contracted HIV during the follow-up period. Our scientific question of interest is to determine the expected incidence of HIV under a hypothetical intervention on the network of sexual partners of each unit i. We note that our dimension reduction assumptions imply that the set of possible realizations of i's partners in a new network draw F_i is restricted to units who were already connected to i at baseline through a common partner, the set we previously denoted as F_i^{0+}. The exposure A_i is defined as a vector of indicators $I(j \in F_i)$ based on the set of all possible partner realizations $j \in F_i^{0+}$. We now assume that $Q_{Y,0}$, the conditional probability of unit i contracting HIV, depends on i's baseline covariates (W_i) and the total number of i's current partners ($nF_i := \sum_{j=1}^{K^+} A_i(j)$). We also assume that $Q_{Y,0}$ depends on the covariates of i's current partners in A_i only as a function of a lower-dimensional summary measure. For example, we suppose that the risk of contracting HIV for unit i also depends on the proportion of the total number of i's partners who had HIV at baseline ($pH_i := (1/nF_i)\sum_{j \in F_i} H_j$), as well as the proportion of i's partners with high-risk as determined by $\delta(W_j) > r_1$, i.e, the summary $pR_i = (1/nF_i)\sum_{j \in F_i} I(\delta(W_j) > r)$, where $\delta(\cdot)$ maps the covariates in W_j into a real line. We now suppose that the exposure summary is defined as $A_i^s := (nF_i, pH_i, pR_i)$, and the baseline summary is defined as $W_i^s := (W_i, \delta(W_j) : j \in F_i^{0+})$. We also consider individual interventions $g_{F_i}^*$ on F_i that replace the observed partners of i in F_i with another set of partners F_i^*. For example, we consider the i-specific intervention $g_{F_i}^*$ that decreases the total number of i's partners by stochastically removing some $j \in F_i$, where this probability of removing a partner $j \in F_i$ can be a function of j's baseline risk-profile $\delta(W_j)$. Such an intervention $g_{F_i}^*$ implies a new exposure set A_i^* and an intervention-specific summary $A_i^{*s} := (nF_i^*, pH_i^*, pR_i^*)$, where $nF_i^* := \sum_{j=1}^{K^+} A_i^*(j)$, $pH_i^* := (1/nF_i^*)\sum_{j \in F_i^*} H_j$ and $pR_i^* := (1/nF_i^*)\sum_{j \in F_i^*} I(\delta(W_j) > r)$. One can now directly apply the TMLE framework from Section 1.4 to estimate the sample-average of the expected outcomes Y_i under such network interventions $g_{F_i^*}$, for $i = 1, \ldots, N$. In particular, we first need to estimate the conditional mixtures $\bar{g}_0(A_i^s|W_i^s)$ and $\bar{g}_0^*(A_i^*|W_i^s)$ using the direct estimation approach, which then allows us to evaluate the clever covariate based on N predictions $[\bar{g}_N^*/\bar{g}_N](A_i^s|W_i^s)$. We then proceed to fit a common-in-i initial model $\bar{Q}_N$ for the regression of Y_i given (A_i^s, W_i), followed by the TMLE update $\bar{Q}_N^*$ for $\bar{Q}_N$.

1.8 Discussion

In this chapter we describe a practical application of the TMLE framework towards the goal of estimation of the sample-average treatment effects in network-dependent data. Our first objective was to assume a realistic semi-parametric data generating model, which reflected the types of between-unit dependence one may encounter in real-life observational network study, for example, when the study units are connected via a social or geographical network. Our approach included a number of statistical assumptions, such as, the assumption of a certain conditional independence of outcomes and the assumption of fixed-dimension summary measures, which allowed us to perform estimation and inference in sample size one problems. Having defined our semi-parametric statistical model $\mathcal{M}$, we also defined our target of estimation $\Psi(P^N)$ as a mapping from the joint distribution P^N on N connected units, for $P^N \in \mathcal{M}$. We then showed in Section 1.3 that this target parameter depends on the joint distribution of the data only as a function of the mixture distribution $\bar{P}$, where $\bar{P}$ was given as a mixture of the unit-specific components P_i of P^N. While this gave us a novel iid interpretation for our target parameter, such mixture mapping representation also implied that our estimation problem was reduced to the problem of estimating the relevant factors of the mixture $\bar{P}$. We have argued that such dependent-data parameters can then be estimated by simply ignoring the dependence between connected observations (e.g., Lemma 1.4.1), suggesting that an entire class of the iid data estimators, such as the iid TMLE algorithm we described in this chapter, may be applied for estimation in dependent data models. That is, we presented the dependent-data TMLE from [63] as a typical iid-data TMLE, with an unusual caveat that our iid-data stochastic intervention $\bar{g}_0^*$ depends on the true distribution of the observed data. We also used the efficient influence curve (EIC) for our target parameter, to provide a simple and consistent estimator of the true asymptotic variance of our TMLE. Our proposed variance estimator took into account the known network structure, making adjustments for correlated outcomes of connected units. We assessed the validity of our inferential framework with a finite sample network simulation study. In particular, the finite sample performance of our proposed TMLE was compared to the parametric G-computation estimator and the IPTW estimator. Lastly, we assessed the finite sample coverage of our estimated asymptotic confidence intervals, e.g., comparing our dependent-data inference to one that ignores the dependence among units (i.e., using the EIC-based variance estimator that treats units as iid). While our simulation results do not necessarily show as low of a coverage as one would expect from the latter iid variance estimator, we nonetheless observed coverage that was consistently below the expected 95%, and was not improved by increasing the sample size. Moreover, we expect coverage for such iid variance estimators to become increasingly worse as one moves towards more realistic network scenarios characterized by denser networks and higher levels of between-unit dependence. We leave this topic to be explored in future simulation studies.

We extended the dependent-data TMLE framework first described in [63] towards the estimation of a much larger class of parameters, such as the direct effect under interference. We have also shown that our framework can be extended to define interventions on the

network itself. In particular, in Section 1.7 we described how one can estimate the post-intervention outcomes for interventions that statically or stochastically modify the initial network structure $\mathbf{F}^0$. Furthermore, we no longer require complete independence of the baseline covariates for conducting valid statistical inference for our TMLE. Finally, we believe our work provides an important proof of concept, demonstrating that estimation and valid statistical inference for dependent data collected from a single network are possible in this large class of semi-parametric models.

We note that the TMLE update $\bar{Q}_N^*$ presented in this chapter differs slightly from the one described in [63] in terms of its suggested parametric submodel fluctuation, $\{\bar{Q}_N(\varepsilon) : \varepsilon\}$, with the latter TMLE update being based on the following parametric submodel: $\text{Logit}\bar{Q}_N(\varepsilon) = \text{Logit}\bar{Q}_N + \varepsilon \bar{g}_0^*/\bar{g}_0$. Both of these fluctuations result in TMLEs with equivalent asymptotic properties, as both updates solve the same empirical score equation. However, the two may differ in their finite sample properties. In particular, the TMLE we present here may be less sensitive to practical positivity violations, while providing similar bias reduction as the TMLE from [63]. We also note that the TMLE update presented here may be less computationally intensive, since it only requires N evaluations of the clever covariate $[\bar{g}_N^*/\bar{g}_N](A_i^s|W_i^s)$, for $i = 1, \ldots, N$. The TMLE algorithm proposed in [63] may require computing $\bar{g}_N^*/\bar{g}_N(a_i^s|W_i^s)$ for every a_i^s in the support of A_i^{*s}, for $i = 1, \ldots, N$ (i.e., all a_i^s such that $g_i^{*s}(a_i^s|W_i^s) > 0$) due to its specific parametric submodel update.

We now note a few possible directions for future research. First, additional simulation studies should explore the performance of our TMLE in more complex networks, such as, networks generated from the preferential attachment model with power law node degree distribution [7] or networks generated under the small world model [135]. Second, it would be of interest to study how our proposed framework may be applied to estimate the change in the observed sample-average outcome when an intervention is applied to another community with a different network structure and different distribution of baseline covariates, a notion known as transportability [91, 8, 90]. Third, it is of scientific interest to explore how our estimation framework can be extended to real-world problems in which data on only a subsample of the full network is available. Moreover, the network structure on the observed units themselves is frequently not fully known, in which case it may be necessary to incorporate the uncertainty introduced by inferring the network structure from the observed data [45]. Finally, future work will investigate the estimation of causal parameters in longitudinal settings where the effect of a single time point intervention can propagate over time through the network, as is typically the case when one describes contagion in social networks [34, 122].

1.9 Acknowledgements

I would like to thank Elizabeth (Betsy) Ogburn for helpful discussions.

1.10 Chapter apendix

Data generating distribution for a simulation study

We implemented a simulation with observed data consisting of N dependent units $\mathbf{O} = (\mathbf{F}, \mathbf{W}, \mathbf{A}, \mathbf{Y})$, where $\mathbf{F} = (F_1, \ldots, F_N)$ is a vector of friends for each unit, $\mathbf{W} = (W_1, \ldots, W_N)$ is a vector of baseline covariates, $\mathbf{A} = (A_1, \ldots, A_N)$ is a vector of binary treatments and $\mathbf{Y} = (Y_1, \ldots, Y_N)$ is a vector of binary outcomes. The data for each unit $i = 1, \ldots, N$ is generated as,

$$\begin{aligned}
W_i &\sim Ber\left(0.35\right) \\
|F_i| &\sim U\left(0, 1, 2\right) \\
F_i \mid |F_i| &\sim Sample_{|F_i|}\left(\{1, \ldots, N\} \setminus i\right) \\
A_i \mid \mathbf{W}, F_i &\sim Ber\left(g_0\left(W_i, (W_j : j \in F_i)\right)\right) \\
Y_i \mid \mathbf{A}, \mathbf{W}, F_i &\sim Ber\left(\bar{Q}_0\left(A_i, W_i, (A_j, W_j : j \in F_i)\right)\right),
\end{aligned}$$

where $F_i \in \{1, \ldots, n\}$ is a set of of unit indices of size $|F_i|$ randomly sampled without replacement, A_i is generated conditionally on the entire vector of baseline covariates $\mathbf{W}$ and Y_i is generated conditionally on $\mathbf{W}$ and all treatment assignments $\mathbf{A}$.

We generate $\mathbf{A}$ with $\mathbf{g}_0$ that depends on unit and unit's friends' baseline covariates,

$$\bar{g}_0\left(W_i, (W_j : j \in F_i)\right) = \text{expit}\left(-1.2 + 1.5 W_i + \sum_{j \in F_i} 0.6 W_j\right),$$

with $\bar{Q}_0$ defined as

$$\bar{Q}_0\left(A_i, W_i, (A_j, W_j : j \in F_i)\right) = \text{expit}\left(-2.5 + 1.5 W_i + 0.5 A_i + \sum_{j \in F_i} 1.5 W_j + \sum_{j \in F_i} 1.5 A_j\right).$$

Notation index

- $\mathbf{O} = (\mathbf{W}, \mathbf{A}, \mathbf{Y})$: the observed data on N units
- $\mathbf{W} = (W_1, \ldots, W_N)$: the observed baseline covariates on N dependent units
- $\mathbf{F} = (F_1, \ldots, F_N)$: the observed network on N units ("network profile")
- $\mathbf{A} = (A_1, \ldots, A_N)$: the observed exposures on N dependent units
- $\mathbf{A}^* = (A_1^*, \ldots, A^*)$: intervened exposures sampled under conditional distribution $\mathbf{G}^*$, given $\mathbf{W}$

- $\mathbf{Y} = (Y_1, \ldots, Y_N)$: the observed outcomes on N connected units
- $W_i^s = w_i^s(\mathbf{W})$, $A_i^s = a_i^s(\mathbf{A}, \mathbf{W})$: fixed-dimension summary measures, dimensionality is the same across all i.
- $\mathbf{O}^s = (\mathbf{W}^s, \mathbf{A}^s, \mathbf{Y})$: observed summary data, where $\mathbf{W}^s = (W_1^s, \ldots, W_N^s)$, $\mathbf{A}^s = (A_1^s, \ldots, A_N^s)$, for $i = 1, \ldots, N$
- $\mathbf{O}^{*\mathbf{s}} = (\mathbf{W}^{\mathbf{s}}, \mathbf{A}^{*\mathbf{s}}, \mathbf{Y}^*)$: summary data sampled from post-intervention distribution under stochastic intervention $\mathbf{g}^*$, with $\mathbf{A}^{*\mathbf{s}} = (A_1^{*s}, \ldots, A_N^{*s})$ and $\mathbf{Y}^* = (Y_1^*, \ldots, Y_N^*)$
- P_0^N: true joint distribution of the observed data $\mathbf{O}$
- p_0^N: true joint density of the observed data $\mathbf{O}$
- $\mathbf{Q}_{\mathbf{W},0}$: true joint distribution of N observed baseline covariates $\mathbf{W}$
- $\mathbf{q}_{\mathbf{W},0}$: true joint density of N observed baseline covariates $\mathbf{W}$
- $\mathbf{G}_0(\mathbf{A} \,|\, \mathbf{W})$: joint conditional distribution for the observed exposures $\mathbf{A}$, given baseline covariates $\mathbf{W}$
- $\mathbf{g}_0(\mathbf{A} \,|\, \mathbf{W})$: joint conditional density for the observed exposures $\mathbf{A}$, given baseline covariates $\mathbf{W}$
- $G_0(A_i \,|\, W_i^s)$: common-in-i conditional distribution for the observed exposure A_i, given the i-specific summary measure of the baseline covariates
- $g_0(A_i \,|\, W_i^s)$: common-in-i conditional density for the observed exposure A_i, given the i-specific summary measure of the baseline covariates
- $\mathbf{G}^*(\mathbf{A}^* \,|\, \mathbf{W})$: user-specified distribution of the intervened network exposure vector $\mathbf{A}^* = (A_1^*, \ldots, A_N^*)$, conditional on baseline covariates $\mathbf{W} = (W_1, \ldots, W_N)$
- $\mathbf{g}^*(\mathbf{A}^* \,|\, \mathbf{W})$: density for the user-specified stochastic intervention of the intervened exposure vector $\mathbf{A}^* = (A_1^*, \ldots, A_N^*)$, conditional on all baseline covariates $\mathbf{W} = (W_1, \ldots, W_N)$
- $\bar{Q}_0(a^s, w^s)$: conditional expectation of the outcome, defined as $E_{P_0^N}[Y_i \,|\, A_i^s = a^s, W_i^s = w^s]$, assumed common across i when conditioned on the same fixed summary measure values a^s, w^s
- $\bar{H}_{i,0}(A_i^s, W_i^s)$: i-specific summary measure distribution for (A_i^s, W_i^s)
- $h_{i,0}(A_i^s, W_i^s)$: i-specific density for the distribution $\bar{H}_{i,0}(A_i^s, W_i^s)$
- $H_{i,0}^*(A_i^s, W_i^s)$: i-specific summary measure distribution for (A_i^{*s}, W_i^s)
- $h_{i,0}^*(A_i^{*s}, W_i^s)$: i-specific density for the distribution $\bar{H}_{i,0}^*(A_i^{*s}, W_i^s)$

- $\bar{h}_0(\bar{A}^s, \bar{W}^s)$: mixture density $1/N \sum_i h_{i,0}$ determined by $\mathbf{g_0}$ and $Q_{\mathbf{W},0}$, with $(\bar{A}^s, \bar{W}^s)$ being a random variable sampled from $\bar{h}_0$
- $\bar{h}_0^*(\bar{A}^{*s}, \bar{W}^s)$: mixture density $1/N \sum_i h_{i,0}^*$ determined by $\mathbf{g}^*$ and $Q_{\mathbf{W},0}$, with $(\bar{A}^{*s}, \bar{W}^s)$ being a random variable sampled from $\bar{h}_0^*$
- $\bar{g}_0(\bar{A}^s \,|\, \bar{W}^s)$: the conditional mixture density implied by factorization

$$\bar{h}_0(\bar{A}^s, \bar{W}^s) = \bar{g}_0(\bar{A}^s \,|\, \bar{W}^s)\bar{Q}_{W^s,0}(\bar{W}^s)$$

- $\bar{g}_0^*(\bar{A}^{*s} \,|\, \bar{W}^s)$: the conditional mixture density implied by factorization

$$\bar{h}^*(\bar{A}^{*s}, \bar{W}^s) = \bar{g}_0^*(\bar{A}^{*s} \,|\, \bar{W}^s)\bar{Q}_{W^s,0}(\bar{W}^s)$$

.

Chapter 2

simcausal R Package for Complex Simulations in Causal Inference

2.1 Introduction

Motivation for simcausal

This chapter describes the **simcausal** package [115], a comprehensive set of tools for the specification and simulation of complex longitudinal data structures to study causal inference methodologies. The package is developed using the R system for statistical computing [96] and is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=simcausal`. Our package is intended to provide a flexible tool to facilitate the process of conducting *transparent* and *reproducible* simulation studies, with a particular emphasis on the types of data and interventions frequently encountered in real-world causal inference problems. For example, our package simplifies the simulation of observational data based on random clinical monitoring to evaluate the effect of time-varying interventions in the presence of time-dependent confounding and sources of selection bias (e.g., informative right censoring). The package provides a novel user-interface that allows concise and intuitive expression of complex functional dependencies between a large number of nodes that may represent time-varying random variables (e.g., repeated measurements over time of the same subject-matter attribute, such as, blood pressure).

Statisticians often rely on simulation studies for assessing the appropriateness and accuracy of different statistical methods [22]. These studies generally help evaluate and uncover potential problems with a method because the statistician knows and controls the true data generating distribution, which remains unknown in a real data study [54]. Hence, a simulation study provides statisticians with a gold standard for evaluating and comparing the performance of different statistical methods. The artificial population data is usually drawn according to the specified model and the statistical procedure is then applied to such data many times. For example, simulations have been applied to evaluate the bias of an estimator [95, 20], study its asymptotic behavior [76], diagnose its sensitivity towards different

modeling assumptions [93, 20], and determine the power of hypothesis tests [124]. Moreover, it may not only be of value to find out that the statistical method works when its postulated assumptions are true, but also to evaluate its robustness towards departures from the required causal and statistical assumptions [32]. These are some of the common reasons why simulation studies are increasingly being used in the medical literature [22, 61, 124, 28]. We also note that careful consideration should be given to a simulation study design [22]. Indeed, simulations are of most value when there is some hope that they are capable of capturing the complexities one might expect to see in real-world data-generating processes. We also argue that careful attention should be paid to the structure and clarity of the simulation code itself, not only to simplify the conduct and presentation of extensive and complex simulation studies, but also to avoid coding errors which may lead to incorrect conclusions and difficulty with reproducing the findings of such a simulation study.

In this package, data can be simulated using a broad range of distributions, such that the resulting data generating distribution falls into a user-specified non-parametric structural equation model (NPSEM) [85, 86, 84]. An NPSEM consists of a set of structural equations, which describe the causal mechanisms for generating independent observations of a user-specified data structure. Each structural equation is used to describe a single variable (call it 'X'), which may be latent or observed. Specifically, the structural equation for X postulates a mechanism in which Nature could have generated X, as a consequence of other endogenous variables' values and a random disturbance (representing the effect of exogenous variables). Thus, defining X in this manner avoids having to make a commitment to a particular parametric family of distributions or specific functional form in which X may relate to other variables. As a result, an NPSEM enforces the separation of the notion of a causal "effect" from its algebraic representation in a particular parametric family (i.e., a coefficient in a linear causal model), and redefines an effect as a 'general capacity to transmit changes among variables' [89, 88]. In particular, the NPSEM framework allows the extension of the capabilities of traditional SEM methods to problems that involve discrete variables, nonlinear dependencies, and heterogeneous treatment effects [35]. The interventions can then be defined by replacing some of the equations in NPSEM with their intervened values, which then defines the counterfactual data.

Our package was developed based on the principles of the NPSEM framework and thus aims to provide the user with a toolkit for specifying and simulating data based on a very large collection of distributions with often nonlinear relationships between the variables. Moreover, **simcausal** is build around the language and the logic of counterfactuals: What would happen if a subject received a different treatment? In other words, **simcausal** also allows for specification and simulation of counterfactual data under various user-specified interventions (e.g., static, dynamic, deterministic, or stochastic), which are referred to as "*actions*". These actions may represent exposure to treatment regimens, the occurrence or non-occurrence of right-censoring events, or of clinical monitoring events (e.g., laboratory measurements based on which treatment decisions may be made). Finally, the package enables the computation of a selected set of "effects" (defined as user-specified features of the distribution of some counterfactual data) that represent common causal quantities of interest,

referred to as *causal target parameters*. For instance, treatment-specific means, the average treatment effects (ATE) (on the multiplicative or additive scale) and coefficients from working marginal structural model (MSM) [101, 77] are a few of the causal target parameters that can be evaluated by the package. The computed value of a particular causal parameter can then serve as the gold standard for evaluating and comparing different estimation methods, e.g., evaluating finite sample bias of an estimator. We note that our package also provides a valuable tool for incorporating and changing various causal independence assumptions and then testing the sensitivity or robustness of the studied statistical methods towards departures from those assumptions.

One of the possible examples of applying **simcausal** in practice includes simulating the types of data collected on subjects in the fields of medicine and public health, e.g., electronic health-records data. Specifically, when one is interested in evaluating the utility and appropriateness of a statistical procedure towards answering causal policy questions about the effects of different interventions on the exposures of interest (e.g., the average effect of a treatment for lowering blood pressure vs. placebo). In addition, our package provides tools for converting simulated and real data between various formats, simplifying the data processing as it may be required by different estimation R packages (e.g., converting longitudinal data from wide to long formats, performing forward imputation on right-censored data). Finally, we note that the **simcausal** package can be a useful instructional tool, since it can elucidate understanding of complex causal concepts [55], for example, using a simulated setting to demonstrate the validity of complex causal identifiability results, showing bias due to unmeasured confounding [37], selection bias [36], and bias due to positivity violations [93]. In summary, these are just a few of the possible practical applications of **simcausal**: (a) Evaluating and comparing the performance of statistical methods and their sensitivity towards departures from specific modeling assumptions; (b) Modeling simulations after real data sets and technically validating an implementation of a novel statistical procedure; (c) Identifying possible issues with statistical algorithms that were not or could not be predicted from theory; and (d) Serving as an instructional tool for understanding complex causal theory in practical simulated settings.

Comparison to other simulation packages

The CRAN system contains several R packages for conducting data simulations with various statistical applications. We reference some of these packages below. Our review is not intended to be exhaustive and we focus on two key aspects in which **simcausal** differ from these other simulation tools.

First, simulations in the **simcausal** package are based on data generating distributions that can be specified via *general* structural equation models. By allowing the specification of a broad range of structural equations, the set of possible distributions available to the analyst for simulating data is meant to be not overly restrictive. For instance, any sampling distribution that is currently available in R or that can be user-defined in the R programming environment can be used for defining the conditional distribution of a node given its parents.

Some of the other R packages rely on alternative approaches for specifying and simulating data. For example, the package **gems** [13] is based on the generalized multistate models, and the package **survsim** [74] is based on the Weibull, log-logistic or log-normal models. Finally, the following R simulation packages rely on linear structural equation models: **lavaan** [108], **lavaan.survey** [82], **sem** [38, 39], **semPLS** [73], **OpenMx** [15, 14] and **simsem** [94]. The latter group of R packages is traditionally described as being based on the **LISREL** model [16]. We note that the purpose and formulation of **LISREL** framework differs from the NPSEM framework that we adopt in **simcausal**, and we use the example in Section 2.3 to help highlight some of the differences. However, describing all the technical details of these two modeling approaches is beyond the scope of this chapter and we refer the reader to the following sources for the additional details: [43, 89, 71, 88, 17, 112].

Second, unlike the **simFrame** package, which is meant as a general object-oriented tool for designing simulation studies, the **simcausal** package is instead tailored to study causal inference methodologies and is particularly suited to investigate problems based on complex longitudinal data structures [101]. Indeed, **simcausal** provides a single pipeline for performing the following common steps frequently encountered in simulation studies from the causal inference literature and described in details later in this chapter: defining the observed data distribution, defining intervention/counterfactual distributions, defining causal parameters, simulating observed and counterfactual data, and evaluating the true value of causal parameters. In addition, the package introduces an intuitive user-interface for specifying complex data-generating distributions to emulate realistic real-world longitudinal data studies characterized by a large number of repeated measurements of the same subject-matter attributes over time. In particular, the **simcausal** package was designed to facilitate the study of causal inference methods for investigating the effects of complex intervention regimens such as dynamic and stochastic interventions (not just the common static and deterministic intervention regimens), and summary measures of these effects defined by (working) marginal structural models. We note, however, that while the package was initially developed for this particular methodological research purpose, its utility can be extended to a broader range of causal inference research, e.g., to perform simulation-based power calculations for informing the design of real-world studies.

Organization of this chapter

The rest of this chapter is organized as follows. In Section 2.2, we provide an overview of the technical details for a typical use of the **simcausal** package. In Section 2.3, we describe a template workflow for a simple simulation study with single time point interventions, while also drawing parallels with the traditional linear SEM framework. In Section 2.4, we describe the use of the package for a more realistic and complex simulation study example based on survival data with repeated measures and dynamic interventions at multiple time points. In Section 2.4, we also apply the **simcausal** package to replicate some of the results of a previously published simulation study by [78, 79]. In Section 2.5, we apply the **simcausal**

package to replicate results of another published simulation study by [68]. We conclude with a discussion in Section 2.6.

2.2 Technical details

NPSEM, causal parameter and causal graph

For the sake of clarity, we limit ourselves to describing a non-parametric structural equation model [NPSEM, [86]] for the observed data collected from a simple single-time point intervention study (no repeated measures on subjects over time) and we note that this NPSEM can be easily extended to longitudinal settings with repeated measures. Suppose that we collect data on baseline covariates, denoted as W, an exposure, denoted as A (e.g. treatment variable), and an outcome of interest, denoted as Y. An NPSEM is a causal model that describes how these variables could be generated from a system of equations, such as: $W = f_W(U_W)$, $A = f_A(W, U_A)$ and $Y = f_Y(W, A, U_Y)$. We note that an NPSEM is defined by unspecified (non-random) functions f_W, f_A, f_Y, and a model on the probability distribution P_U of random "disturbances" $U = (U_W, U_A, U_Y)$. These equations are non-parametric in the sense that they make no specific statement about the functional form of f_W, f_A, f_Y. We define the *observed data*[1]as $O = (W, A, Y)$, and we note that the allowed set of probability distributions for O is referred to as the the *statistical* model and it is implied by the *causal* model encoded by the above NPSEM (i.e., by the choice of f and the choice of the distribution P_U). We also note that every parametric data-generating distribution defined in the **simcausal** package can be described as an instance of a distribution in some NPSEM. Such NPSEM encodes the independence assumptions between the endogenous variables. For instance, the NPSEM described above assumes that the exposure A can depend on all baseline variables W. As another example, suppose that (W, A, Y) were collected from a clinical trial in which the exposure A was assigned at random. In this case, A is independent of W, an assumption that can be encoded in the above NPSEM by removing W from the above equation f_A as follows: $A = f_A(U_A)$.

The NPSEM also implicitly encodes the definition of counterfactual variables, i.e., variables which would result from some particular interventions on a set of endogenous variables. For example, the NPSEM can be modified as follows: $W = f_W(U_W)$, $A = a$, $Y_a = f_Y(W, a, U_Y)$, where the equation for W was kept unchanged, A was set to a known constant a and Y_a denotes the counterfactual outcome under an intervention that sets $A = a$. In this chapter, we will refer to (W, a, Y_a) as *counterfactual data* and we define our target causal parameter as a function of such counterfactual data distribution, resulting from one or more exposure intervention "a". For example, the average treatment effect (ATE) can be expressed as $E[Y_1 - Y_0]$. The fundamental feature of the causal parameter defined in this manner is that it remains a well-defined quantity under any probability distribution P_U

[1]We use the term "observed data" to designate the collection of all non-latent endogenous variables. The term "observed data" is meant to be opposed to the "counterfactual data" defined in the next paragraph.

for the disturbances and any choice of functions f, a notion which we also highlight with examples in Section 2.3.

Furthermore, suppose our goal is to evaluate the effect of the exposure with more than two levels (e.g., categorical or time-varying A), in which case we could evaluate the above ATE for any two possible combinations of different exposure levels. We could also undertake an equivalent approach and characterize all such contrast with a saturated model for the mean counterfactual outcome ($E(Y_a)$), as indexed by the exposure levels a of interest. For example, for an exposure with levels $a \in \{0,1,2\}$, we may use the following saturated MSM with three parameters: $E(Y_a) = \alpha_0 + \alpha_1 I(a=1) + \alpha_2 I(a=2)$. This model then implies that each possible contrast (ATE) can be recovered as a function of $\alpha = (\alpha_0, \alpha_1, \alpha_2)$, e.g., $E(Y_1 - Y_0) = \alpha_1$. However, this approach becomes problematic when dealing with small sample datasets and high dimensional or continuous exposures. That is, suppose our goal is to characterize the entire causal function of a given by $\{E(Y_a) : a \in \mathcal{A}\}$, where $\mathcal{A}$ represents the support of a highly dimensional or continuous A. An alternative approach is to approximate the true causal function $\{E(Y_a) : a \in \mathcal{A}\}$ with some low-dimensional *working* marginal structural model $m(a|\alpha)$. For example, one may define the working MSM as the following linear model: $m(a|\alpha) = \alpha_0 + \alpha_1 a + \alpha_2 a^2$. Note, however, that the term "working MSM" implies that we are not assuming $E(Y_a) = m(a|\alpha)$, but instead we are defining our causal parameter (α) as the best parametric approximation of the true function $E(Y_a)$ with $m(a|\alpha)$. That is, such a working MSM made no assumptions about the true functional form of $E(Y_a)$ and thus made no additional assumptions about the distribution of U and the functions f, beyond those already implied by the NPSEM (e.g., independence of (U_W, U_A, U_Y)). We also refer to [77] for additional details and examples of working MSMs. Also note that the concept of such working MSMs is easily extended to arbitrary functions, e.g., we could define $m(a|\alpha)$ as an *expit* function when the outcome Y is binary.

We note that the above NPSEM can be equivalently represented as a Directed Acyclic Graph (DAG) [85], such as the one in Figure 2.1 (left), by drawing arrows from causes to their effects. Links in this DAG can be of two kinds: those that involve unmeasured quantities are represented by dashed arrows and those that only involve measured quantities by solid arrows. We note that each endogenous node in Figure 2.1 represents a single equation in the above NPSEM. The causal assumptions in such a DAG are conveyed by the missing arrows, i.e., in our second example of the NPSEM, the absence of a variable W from the right-hand side of the equation for $A = f_A(U_A)$ would correspond with no direct arrow between W and A. The disturbances U (also referred to as 'errors') are enclosed in circles in the diagram on the left because they represent unobserved (latent) factors that the modeler decides to keep unexplained. When the error terms (U_W, U_A, U_Y) are assumed to be independent, the often-used convention is to remove them from the causal DAG [88], as shown in Figure 2.1 (right), with the implication that each of the remaining variables is subject to the influence of its own independent error. This is also precisely how the function `plotDAG` of the **simcausal** package will plot the diagram of the user-specified SEM, that is, omitting the implied independent errors that influence each user-defined latent and endogenous node. We also refer to the examples in Section 2.3 for illustrations of this functionality of **simcausal**.

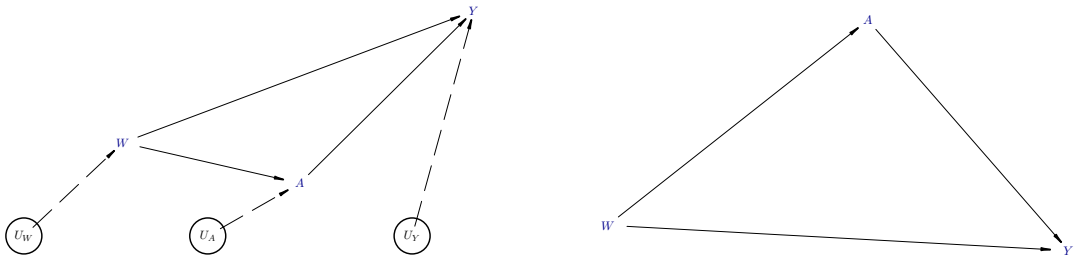

Figure 2.1: Two alternative ways to graphically represent the same structural equation model (SEM) using directed acyclic graphs (DAGs). The left figure shows the independent (latent) errors, while the right figure doesn't.

$$
\begin{array}{ccc}
D = \texttt{DAG.empty}() & & \\
\downarrow & & \\
D = D + \texttt{node}(\ldots) + \texttt{node}(\ldots) & & \\
\downarrow & & \\
D = \texttt{set.DAG}(D) & \longrightarrow & \texttt{sim}(D) \\
\downarrow & & \\
D = D + \texttt{action}(\ldots) & \longrightarrow & \texttt{sim}(D, \texttt{actions} = \ldots) \\
\downarrow & & \\
D = \texttt{set.targetE}(D, \ldots) & & \\
D = \texttt{set.targetMSM}(D, \ldots) & & \\
\downarrow & & \\
\texttt{eval.target}(D) & &
\end{array}
$$

Figure 2.2: Schematic of **simcausal** routines and the order in which one would usually call such routines in a typical simulation study.

We note that **simcausal** was designed to facilitate simulations from NPSEM with mutually independent disturbances. However, we also note that one can use **simcausal** to simulate dependent errors (U) with an arbitrary correlation structure using one of the following methods: a) Sample U jointly using a user-specified multivariate distribution with a specific correlation structure, e.g., multivariate normal or copula (see the documentation and examples for the `node` function); b) Create a common (also latent) parent that has a direct effect of all three variables in U (see the example in Section 2.3; or c) Perform Cholesky decomposition of the covariance matrix Σ for a multivariate normal $N(\mu, \Sigma)$, then generate *correlated* (U_W, U_A, U_Y) distributed as $N(\mu, \Sigma)$ based on the previously sampled independent standard normal variables (see the example in Section 2.5).

The workflow

Data structures. The following most common types of output are produced by the package.

parameterized causal DAG model - object that specifies the structural equation model, along with interventions and the causal target parameter of interest.

observed data - data simulated from the (pre-intervention) distribution specified by the structural equation model.

counterfactual data - data simulated from one or more post-intervention distributions defined by actions on the structural equation model.

causal target parameter - the true value of the causal target parameter evaluated with counterfactual data.

Routines. The following routines, also outlined in Figure 2.2, will be generally invoked by a user, in the same order as presented below.

`DAG.empty` initiates an empty `DAG` object that contains no nodes.

`node` defines a node in the structural equation model and its conditional distribution, i.e., the outcome of one equation in the structural equation model and the formula that links the outcome value to that of earlier covariates, referred to as parent nodes. A call to `node` can specify either a single node or multiple nodes at once, with `name` and `distr` being the only required arguments. To specify multiple nodes with a single `node` call, one must also provide an indexing vector of integers as an argument `t`. In this case, each node shares the same name, but is indexed by distinct values in `t`. The simultaneous specification of multiple nodes is particularly relevant for providing a shorthand syntax for defining a time-varying covariate, i.e., for defining repeated measurements over time of the same subject-matter attribute, as shown in the example in Section 2.4.

`add.nodes` **or** `D + node` provide two equivalent ways of growing the structural equation model by adding new nodes and their conditional distributions. Informally, these routines are intended to be used to sequentially populate a `DAG` object with all the structural equations that make up the causal model of interest. See Sections 2.3 and 2.4 for examples.

`set.DAG` locks the `DAG` object in the sense that no additional nodes can be subsequently added to the structural equation model. In addition, this routine performs several consistency checks of the user-populated `DAG` object. In particular, the routine attempts to simulate observations to verify that all conditional distributions in the `DAG` object are well-defined.

sim simulates independent and identically distributed (iid) observations of the complete node sequence defined by a DAG object. The output dataset is stored as a data.frame and is referred to as the *observed data*. It can be structured in one of two formats, as discussed in Section 2.4.

add.action or D + action provides two equivalent ways to define one or more actions. An action modifies the conditional distribution of one or more nodes of the structural equation model. The resulting data generating distribution is referred to as the post-intervention distribution. It is saved in the DAG object alongside the original structural equation model. See Sections 2.3 and 2.4 for examples.

sim(..., actions = ...) can also be used for simulating independent observations from one or more post-intervention distributions, as specified by the actions argument. The resulting output is a named list of data.frame objects, collectively referred to as the *counterfactual data*. The number of data.frame objects in this list is equal to the number of post-intervention distributions specified in the actions argument, where each data.frame object is an iid sample from a particular post-intervention distribution.

set.targetE and set.targetMSM define two distinct types of target causal parameters. The output from these routines is the input DAG object with the definition of the target causal parameter saved alongside the interventions. See Sections 2.3 and 2.4 for examples defining various target parameters.

eval.target evaluates the causal parameter of interest using simulated counterfactual data. As input, it can take previously simulated counterfactual data (i.e., the output of a call to the sim(..., actions = ...) function) or, alternatively, the user can specify the sample size n, based on which counterfactual data will be simulated first.

Specifying a structural equation model

The **simcausal** package encodes a structural equation model using a DAG object. The DAG object is a collection of nodes, each node represented by a DAG.node object that captures a single equation of the structural equation model. DAG.node objects are created by calling the node function. When the node function is used to simultaneously define multiple nodes, these nodes share the same name, but must be indexed by distinct user-specified integer values of the time variable t, as shown in the example in Section 2.4. We will refer to a collection of nodes defined simultaneously in this manner as a *time-varying node* and we will refer to each node of such a collection as a measurement at a specific time point.

Each node is usually added to a growing DAG object by using either the add.nodes function or equivalently the '+' function, as shown in the example in Sections 2.3 and 2.4. Each new node added to a DAG object must be uniquely identified by its name or the combination of a name and a value for the time variable argument t.

The user may explicitly specify the temporal ordering of each node using the `order` argument of the `node()` function. However, if this argument is omitted, the `add.nodes` function assigns the temporal ordering to a node by using the actual order in which this node was added to the `DAG` object and, if applicable, the value of the time variable that indexes this node (earlier added nodes receive a lower order value, compared to those that are added later; nodes with a lower value for the `t` argument receive a lower order value, compared to those with a higher value of `t`).

The `node` function also defines the conditional distribution of a node, given its parents, with a combination of the sampling distribution specified by the `distr` argument and the distributional parameters specified as additional named arguments to the `node()` function. This `distr` argument can be set to the name of any R function that accepts an integer argument named `n` and returns a vector of size `n`. Examples of such distribution functions are provided in Section 2.3.

The distributional parameters are specified as additional named arguments of the `node()` function and can be either constants or some summary measures of the parent nodes. Their values can be set to any evaluable R expressions that may reference any standard or user-specified R function, and also, may invoke a novel and intuitive shorthand syntax for referencing specific measurements of time-varying parent nodes, i.e., nodes identified by the combination of a node name and a time point value `t`. The syntax for identifying specific measurements of time-varying nodes is based on a re-purposed R square-bracket vector subsetting function `'['`: e.g., writing the expression `sum(A[0:5])`, where `A` is the name of a previously defined time-varying node, defines the summary measure that is the sum of the node values over time points `t = 0`,...,`5`. This syntax may also be invoked to simultaneously define the conditional distribution of the measurements of a time-varying node over multiple time points `t` at once. For example, defining the conditional distribution of a time-varying node with the R expression `sum(A[max(0, t - 5):t]) + t` will resolve to different node formulas for each measurement of the time-varying node, depending on the value of `t`:

1. `A[0]` at `t = 0`;
2. `sum(A[0:1]) + 1` at `t = 1`, ..., `sum(A[0:5]) + 5` at `t = 5`;
3. `sum(A[1:6]) + 6` at `t = 6`, ..., `sum(A[5:10]) + 10` at `t = 10`.

Concrete applications of this syntax are described in Section 2.4, as well as in the documentation of the `node()` function (`?node`).

Note that the user can also define a causal model with one or more nodes that represent the occurrence of *end of follow-up* (EFU) events (e.g., right-censoring events or failure events of interest). Such nodes are defined by calling the `node()` function with the `EFU` argument being set to `TRUE`. The EFU nodes encode binary random variables whose value of 1 indicates that, by default, all of the subsequent nodes (i.e., nodes with a higher temporal order value) are to be replaced with a constant `NA` (missing) value. As an alternative, the user may choose to impute missing values for the time-varying node that represents the failure event

of interest using the *last time point value carried forward* (LTCF) imputation method. This imputation procedure consists in replacing missing values for measurements of a time-varying node at time points `t` after an end of follow-up event with its last known measurement value prior to the occurrence of an end of follow-up event. Additional details about this imputation procedure are provided in the **simcausal** package vignette Section 4.6 [115].

Finally, we note that the package includes pre-written wrapper functions for random sampling from some commonly employed distributions. These routines can be passed directly to the `distr` argument of the node function with the relevant distributional parameters on which they depend. These built-in functions can be listed at any time by calling `distr.list()`. In particular, the routines `"rbern"`, `"rconst"`, and `"rcat.b1"` can be used for specifying a Bernoulli distribution, a degenerate distribution (constant at a given value), and a categorical distribution, respectively. One can also use any of the standard random generating R functions, e.g., `"rnorm"` for sampling from the normal distribution and `"runif"` for sampling from the uniform distribution, as demonstrated in Sections 2.3 and 2.3.

Specifying interventions

An intervention regimen (also referred to as action regimen) is defined as a sequence of conditional distributions that replace the original distributions of a subset of nodes in a `DAG` object. To specify an intervention regimen, the user must identify the set of nodes to be intervened upon and provide new node distributions for them. The user may define a static, dynamic, deterministic or stochastic intervention on any given node, depending on the type of distributions specified. A deterministic static intervention is characterized by replacing a node distribution with a degenerate distribution such that the node takes on a constant value. A deterministic dynamic intervention is characterized by a conditional degenerate distribution such that the node takes on a value that is only a function of the values of its parents (i.e., a decision rule). A stochastic intervention is characterized by a non-degenerate conditional distribution. A stochastic intervention is dynamic if it is characterized by a non-degenerate conditional distribution that is defined as a function of the parent nodes and it is static otherwise. Note that a particular intervention may span different types of nodes and consist of different types of distributions, e.g., an intervention on a monitoring node can be static, while the intervention on a treatment node from the same structural equation model may be dynamic.

To define an intervention the user must call `D + action(A, nodes = B)` (or equivalently `add.action(D, A, nodes = B)`), where `D` is a `DAG` object, `A` is a unique character string that represents the intervention name, and `B` is a list of `DAG.node` objects defining the intervention regimen. To construct `B` the user must first aggregate the output from one or more calls to `node` (using `c(..., ...)`), with the `name` argument of the `node` function call set to node names that already exist in the locked `DAG` object `D`. The example in Section 2.4 demonstrates this functionality. Alternatively, repeated calls to `add.action` or `D+action` with the same intervention name, e.g., `A = "A1"`, allow the incremental definition of an intervention regimen by passing each time a different `node` object, enabling iterative build-up of the collection `B`

of the intervened nodes that define the intervention regimen. Note, however, that by calling `D + action` or `add.action(D, ...)` with a new action name, e.g., `action("A2", ...)`, the user initiates the definition of a new intervention regimen.

Specifying a target causal parameter

The causal parameter of interest (possibly a vector) is defined by either calling the function `set.targetE` or `set.targetMSM`. The function `set.targetE` defines causal parameters as the expected value(s) of `DAG` node(s) under one post-intervention distribution or the contrast of such expected value(s) from two post-intervention distributions. The function `set.targetMSM` defines causal parameters based on a *working* marginal structural model [77]. In both cases, the true value of the causal parameter is defined by one or several post-intervention distributions and can thus be approximated using counterfactual data.

The following types of causal parameters can be defined with the function `set.targetE`:

- The expectation of an outcome node under an intervention regimen denoted by d, where the outcome under d is denoted by Y_d. This parameter can be naturally generalized to a vector of measurements of a time-varying node, i.e., the collection of nodes $Y_d(t)$ sharing the same name, but indexed by distinct time points `t` that represents a sequence of repeated measurements of the same attribute (e.g., a CD4 count or the indicator of past occurrence of a given failure event):

$$E(Y_d) \text{ or } (E(Y_d(t)))_{t=0,1,\ldots}.$$

- The difference between two expectations of an outcome node under two interventions, d_1 and d_0. This parameter can also be naturally generalized to a vector of measurements of a time-varying node:

$$E(Y_{d_1}) - E(Y_{d_0}) \text{ or } (E(Y_{d_1}(t)) - E(Y_{d_0}(t)))_{t=0,1,\ldots}.$$

- The ratio of two expectations of an outcome node under two interventions. This parameter can also be naturally generalized to a vector of measurements of a time-varying node:

$$\frac{E(Y_{d_1})}{E(Y_{d_0})} \text{ or } \left(\frac{E(Y_{d_1}(t))}{E(Y_{d_0}(t))}\right)_{t=0,1,\ldots}.$$

Note that if the `DAG` object contains nodes of type `EFU = TRUE` other than the outcome nodes of interest $Y_d(t)$, the target parameter must be defined by intervention regimens that set all such nodes that precede all outcomes of interest $Y_d(t)$ to 0. Also note that with such intervention regimens, if the outcome node is time-varying of type `EFU = TRUE` then the nodes $Y_d(t)$ remain well defined (equal to 1) even after the time point when the simulated value for the outcome jumps to 1 for the first time. The nodes $Y_d(t)$ can then be interpreted

as indicators of past failures in the absence of right-censoring events. The specification of these target parameters is covered with examples in Sections 2.3 and 2.4.

When the definition of the target parameter is based on a working marginal structural model, the vector of coefficients (denoted by α) of the working model defines the target parameter. The definition of these coefficients relies on the specification of a particular weighting function when the working model is not a correct model (see [77] for details). This weighting function is set to the constant function of 1 in this package. The corresponding true value of the coefficients α can then be approximated by running a standard (unweighted) regression routines applied to simulated counterfactual data observations. The following types of working models, denoted by $m()$, can be defined with the function `set.targetMSM`:

- The working linear or logistic model for the expectation of one outcome node under intervention d, possibly conditional on baseline node(s) V, where a baseline node is any node preceding the earliest node that is intervened upon, i.e., $E(Y_d \mid V)$:

  $$m(d, V \mid \alpha).$$

  Such a working model can, for example, be used to evaluate the effects of HIV treatment regimens on the mean CD4 count measured at one point in time.

- The working linear or logistic model for the expectation vector of measurements of a time-varying outcome node, possibly conditional on baseline node(s) V, i.e., $E(Y_d(t) \mid V)$:

  $$m(t, d, V \mid \alpha), \text{ for } t = 0, 1, \ldots.$$

  Such a working model can, for example, be used to evaluate the effects of HIV treatment regimens on survival probabilities over time.

- The logistic working model for discrete-time hazards, i.e., for the probabilities that a measurement of a time-varying outcome node of type `EFU=TRUE` is equal to 1 under intervention d, given that the previous measurement of the time-varying outcome node under intervention d is equal to 0, possibly conditional on baseline node(s) V, i.e., $E(Y_d(t) \mid Y_d(t-1) = 0, V)$:

  $$m(t, d, V), \text{ for } t = 0, 1, \ldots.$$

  Such a working model can, for example, be used to evaluate the effects of HIV treatment regimens on discrete-time hazards of death over time.

Examples of the specification of the above target parameters are provided in Sections 2.3 and 2.4. As shown above, the working MSM formula $m()$ can be a function of t, V and d, where d is a unique identifier of each intervention regimen. In Sections 2.3 and 2.4 we describe in detail how to specify such identifiers for d as part of the `action` function call. Also note that the working MSM formula, m, may reference time-varying nodes using the square-bracket syntax introduced in Section 2.2, as long as all such instances are embedded within the syntax `S(...)`. Example use of this syntax is provided in Section 2.4 (Example 2 of `set.targetMSM`).

Simulating data and evaluating the target causal parameter

The **simcausal** package can simulate two types of data: 1) observed data, sampled from the (pre-intervention) distribution specified by the structural equation model and 2) counterfactual data, sampled from one or more post-intervention distributions defined by actions on the structural equation model. Both types of data are simulated by invoking the `sim` function and the user can set the seed for the random number generator using the argument `rndseed`. The examples showing how to simulate observed data are provided in Sections 2.3 and 2.4, whereas the examples showing how to simulate counterfactual data are provided in Sections 2.3 and 2.4.

We note that two types of structural equation models can be encoded with the `DAG` object: 1) models where some or all nodes are defined by specifying the "time" argument `t` to the `node` function, or 2) models where the argument `t` is not used for any of the nodes. For the first type of structural equation models, the simulated data can be structured in either *long* or *wide* formats. A dataset is considered to be in wide format when each simulated observation of the complete sequence of nodes is represented by only one row of data, with each time-varying node represented by columns spanning distinct values of `t`. In contrast, for a dataset in long format, each simulated observation is typically represented by multiple rows of data indexed by distinct values of `t` and each time-varying node represented by a single column. The format of the output data is controlled by setting the argument `wide` of the `sim` function to `TRUE` or `FALSE`. The default setting for `sim` is to simulate data in wide format, i.e., `wide = TRUE`. An example describing these two formats is provided in Section 2.4.

In addition, as previously described, for nodes representing the occurrence of end of follow-up events (i.e., censoring or outcome nodes declared with `EFU = TRUE`), the value of 1 indicates that, during data simulation, by default, all values of subsequent nodes (including the outcome nodes) are set to missing (`NA`). To instead impute these missing values after a particular end of follow-up event occurs (typically the outcome event) with the *last time point value carried forward* (LTCF) method, the user must set the argument `LTCF` of the `sim` function to the name of the `EFU`-type node that represents the end of follow-up event of interest. This will result in carrying forward the last observed measurement value for all time-varying nodes, after the value of the EFU node whose name is specified by the `LTCF` argument is observed to be 1. For additional details see the package documentation for the function `sim`.

In the last step of a typical workflow, the function `eval.target` is generally invoked for estimation of the true value of a previously defined target causal parameter. The true value is estimated using counterfactual data simulated from post-intervention distributions. The function `eval.target` can be called with either previously simulated counterfactual data, specified by the argument `data` or a sample size value, specified by the argument `n`. In the latter case, counterfactual data with the user-specified sample size will be simulated first.

2.3 Simulation study with single time point interventions.

The following examples describe a typical workflow for specifying a structural equation model, defining various interventions, simulating observed and counterfactual data, and calculating various causal target parameters. The structural equation model chosen here illustrates a common point treatment problem in which one is interested in evaluating the effect of an intervention on one treatment node on a single outcome node using observational data with confounding by baseline covariates. In addition, these examples demonstrate the plotting functionality of the **simcausal** package that builds upon the **igraph** R package [29] to visualize the Directed Acyclic Graph (DAG) [85, 86, 84] implied by the structural equation model encoded in the `DAG` object.

We also undertake an approach similar to the one described in [35] and use the following examples to highlight some of the differences between the non-parametric structural equation models [86] and the traditional linear structural equation models based on the **LISREL** framework [16]. Many traditional applications of structural equation modeling are devoted to addressing the problem of the measurement in the exposure, and more precisely, to address problems in which the true exposure of interest is a latent variable, such as talent, motivation or political climate that cannot be observed directly, but that is instead measured via some noisy and correlated proxies. Hence, the **LISREL** framework is frequently applied to formally assess the causal effects of such latent variables. However, the primary intended goal of **simcausal** is not to simulate such measurement error data, even though one could adapt **simcausal** for that purpose. Instead, our package specifically focuses on data simulation for the purpose of evaluating estimation methods for assessing the effect of exposures that can be observed directly. Additionally, one may also use **simcausal** to simulate data problems with latent variables that might impact the observed exposures of interest.

Specifying parametric structural equation models in simcausal

Suppose that we want to simulate data that could be generated in a hypothetical study evaluating the effect of receiving school vouchers on mean test scores based on a sample of students. We start by assuming that a latent covariate I represents the level of subject's true and unobserved intelligence, where I is categorical and its distribution is defined by the node named `"I"` in the code example below. We also assume that I directly influences the values of the three observed baseline covariates $W = (W_1, W_2, W_3)$ (nodes `"W1"`, `"W2"` and `"W3"` below) and we define the distribution of each W conditional on I. That is, the observed baseline covariates in W will be correlated, since all three depend on a common and latent parent I. We now let A (node `"A"` below) define the observed binary exposure (receiving school vouchers), where the probability of success for A is defined as the following logit-linear function[2]of W:

$$logit(P(A = 1|W)) = \alpha_0 + \gamma_A W,$$

for $W = (W_1, W_2, W_3)^t$, $\alpha_0 = 4.2$ and $\gamma_A = (-0.5, 0.1, 0.2)$.

That is, the above model assumes that A is directly influenced by the observed variable W, while the latent I has no direct influence on A. We also emphasize that we want to study the effect of intervening on the observed variable(s), such as A, whereas in the traditional measurement error model the focus might have been on modeling the effect of the latent varible I on some observed outcome(s). The following example code defines the distributions of (I, W, A). Specifically, we use the pre-defined R functions `rcat.b1`, `rnorm`, `runif` and `rbern` to define the latent categorical node `I`, normal node `W1`, uniform node `W2` and Bernoulli nodes `W3` and `A`, respectively[3]. We also note that implicit in the specification of these nodes is the specification of independent exogenous errors (disturbances), whose distributions are defined by the `distr` arguments as shown below.

```
library("simcausal")
D <- DAG.empty()
D <- D +
  node("I", distr = "rcat.b1",
    probs = c(0.1, 0.2, 0.2, 0.2, 0.1, 0.1, 0.1)) +
  node("W1", distr = "rnorm",
    mean = ifelse(I == 1, 0, ifelse(I == 2, 3, 10)) + 0.6 * I, sd = 1) +
  node("W2", distr = "runif",
    min = 0.025*I, max = 0.7*I) +
  node("W3", distr = "rbern",
    prob = plogis(-0.5 + 0.7*W1 + 0.3*W2 - 0.2*I)) +
  node("A", distr = "rbern",
    prob = plogis(+4.2 - 0.5*W1 + 0.1*W2 + 0.2*W3))
```

Similarly, we assume that the outcome Y is influenced by an independent latent error $U_Y \sim N(0,1)$, and we use the following code example to show how one might explicitly define U_Y using a node named `"U.Y"`[4]:

```
D <- D + node("U.Y", distr = "rnorm", mean = 0, sd = 1)
```

[2] $logit(x) = \log[x/(1-x)]$

[3] For details and examples on writing sampling functions for arbitrary distributions see Section 2.3. We also refer to Section 2.3 for a description on how to specify node formulas (distributional parameters), such as, the R expressions specified by the `probs`, `mean`, `sd`, `min`, `max` and `prob` arguments to `node` function.

[4] In **simcausal**, such disturbances would typically be defined implicitly as representing mutually independent exogenous variables, as shown in the previous examples of node specification. We can however also define them explicitly as endogenous variables. For example, this can be done for the purpose of defining non-independent error terms. For simplicity here, we demonstrate how such error terms can be defined explicitly and refer the reader to the previous Section 2.2 and help files for a descriptions of 3 alternative methods for defining non-independent errors.

The following example defines the outcome Y (node named `"Y"`) by using the following linear structural equation:

$$Y = \beta_0 + \beta_1 A + \beta_2 I + \gamma_Y W + U_Y,$$

where $\beta_0 = -0.5$, $\beta_1 = 1.2$, $\beta_2 = 0.2$ and $\gamma_Y = (0.1, 0.3, 0.2)$.

Note that in this example, we are assuming that the effect of exposure A on Y is the same for every strata of W and I (i.e., *homogeneous* treatment effect). We also note that the distribution of the node `Y` is defined below as *degenerate* (`distr = "rconst"`), since we explicitly define its error term with the above node `U.Y`. That is, the following example uses a pre-defined R function `rconst`, which puts mass one on the value of the `node` function argument `const`:

```
D <- D + node("Y", distr = "rconst",
    const = -0.5 + 1.2*A + 0.2*I + 0.1*W1 + 0.3*W2 + 0.2*W3 + U.Y)
```

Note that the names of all user-defined endogenous latent nodes must be specified within the `set.DAG` function via the argument `latent.v`, as shown in this example:

```
Dset1 <- set.DAG(D, latent.v = c("I", "U.Y"))
```

Running the code above results in implicitly assigning a sampling order (temporal order) to each node - based on the order in which the nodes were added to the `DAG` object `D`. Alternatively, one can use the optional `node()` argument order to explicitly specify the integer value of the sampling order of each node, as described in more detail in the documentation for the node function. The resulting internal representation of the structural equation model encoded by the `DAG` object `Dset1` can be examined as follows:

```
str(Dset1)
```

In the example above, we are interested in the causal target parameter defined as the average treatment effect (ATE) of school vouchers on mean test scores, which is generally defined in the NPSEM framework as $E(Y_1 - Y_0)$. Analytically, one can show that in the simple SEM defined above, the ATE is equal to the coefficient β_1 [88].

Our example so far illustrates a scenario typical of the linear SEM literature in which the effect of interest corresponds with a coefficient from one of the structural equations. We now illustrate other more complex scenarios in which the effect of interest (ATE) is not equal to one particular structural equation coefficient. In the following example, we modify the above SEM for Y and allow for the effect of treatment on Y to vary by strata of W_3:

$$Y = \beta_0 + \beta_1 A + \beta_1^*(AW_3) + \beta_2 I + \gamma_Y W + U_Y,$$

where $\beta_0 = -0.5$, $\beta_1 = 1.2$, $\beta_1^* = -0.5$, $\beta_2 = 0.2$ and $\gamma_Y = (0.2, 0.2, 0.2)$. Note that in this example we moved away from the classical linear structural model for Y, specifically, we

allowed for the causal effect of A on Y to vary by subject depending on their value of W_3. Finally, we note that whenever the node named `"Y"` is added again to the same `DAG` object `D`, **simcausal** automatically overwrites the previously defined distribution of Y with the one given by the new `node` function call, as demonstrated below.

```
D <- D + node("Y", distr = "rconst",
    const = -0.5 + 1.2*A - 0.5*(A * W3) + 0.2*I + 0.2*(W1 + W2 + W3) + U.Y)
Dset2 <- set.DAG(D, latent.v = c("I", "U.Y"))
```

Note that for the above data generating distribution specified by the object `Dset2`, the ATE ($E(Y_1 - Y_0)$) is no longer equal to β_1, but is rather equal to $\beta_1 + \beta_1^* E(W_3)$ (proof not shown, but easily derived by following the same logic as in the previous example).

For our final example shown below, we re-define Y as a nonlinear function of the same parent nodes used in the previous two examples:

$$\begin{aligned} Y =& \beta_1 A + \beta_2(W_1^2 + W_2^3/10 + W_3) + \beta_3|U_Y| + \beta_4 I^2 \\ &+ \beta_5|\frac{1}{\sin(U_Y W_2 + A)}|h_Y(U_Y, W) + \beta_6(1 - h_Y(U_Y, W)), \end{aligned}$$

where $h_Y(U_Y, W) = I(|1/\sin(U_Y W_2)| \leq 10)$, $\beta_1 = 1.2$, $\beta_2 = 0.05$, $\beta_3 = 0.7$, $\beta_4 = 0.002$, $\beta_5 = 0.02$ and $\beta_6 = 5$. Note that in this model for the outcome Y, the analytic derivation of the ATE becomes intractable. However, one can use **simcausal** to find a Monte-Carlo approximation of the ATE from simulated counterfactual data, as shown in Section 2.3.

```
D <- D + node("Y", distr = "rconst",
    const =
      +1.2*A + 0.05*(W1^2 + W2^3 / 10 + W3) + 0.7*abs(U.Y) + 0.002*I^2 +
      +0.02*abs(1 / sin(U.Y * W2 + A)) * (abs(1/sin(U.Y * W2)) <= 10) +
      +5*(abs(1/sin(U.Y * W2)) > 10))
Dset3 <- set.DAG(D, latent.v = c("I", "U.Y"))
```

We note that all three of the above structural equations for Y depend on exactly the same variables, namely, (A, W, I). Therefore, the three parameterizations of the SEM specified by the above objects `Dset1`, `Dset2` and `Dset3` are represented by the same DAG in Figure 2.3. The DAG in Figure 2.3 was automatically generated by calling the function `plotDAG`. The plotting is accomplished by using the visualization functionality from the **igraph** package [29]. The directional arrows (solid and dashed) represent the functional dependencies in the structural equation model. Specifically, the node of origin of each arrow is an extracted node name from the *node formula(s)*. The user-specified latent nodes are surrounded by circles, and each functional dependency that originates at a latent node is displayed via a dashed directional arrow[5].

[5]Note that the appearance of the resulting diagram can be customized with additional arguments, as demonstrated in the **simcausal** package vignette [115].

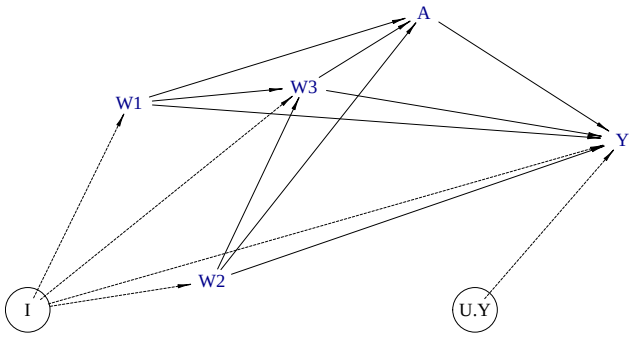

Figure 2.3: Graphical representation of the structural equation model using a DAG, where the latent nodes `I` and `U.Y` are enclosed in circles.

The above alternative examples for specifying the outcome variable Y also demonstrate how **simcausal** can be applied for defining a variety of functional and distributional relationships between the model variables, including those that can be specified by the traditional linear structural equation models. We have also demonstrated that our package can be used for defining the SEM with endogenous latent variables. The above examples also highlight the merit of defining the target causal parameters in a way that remains meaningful for any parametric specification of the SEM. As we demonstrate in Section 2.3 below, our package provides exactly this type of functionality, allowing the user to define and evaluate various causal target parameters as functions of the counterfactual data distribution.

Simulating observed data (`sim`)

Simulating observed data is accomplished by calling the function `sim` and specifying its arguments `DAG` and `n` that indicate the causal model and sample size of interest. Below is an example of how to simulate an observed dataset with 10,000 observations using the causal model defined in the previous section. The output is a `data.frame` object.

```
Odat <- sim(DAG = Dset3, n = 10000, rndseed = 123)
```

The format of the output dataset is easily understood by examining the first row of the `data.frame` returned by the `sim` function. Note that the latent variables '`I`' and '`U.Y`' are absent from the simulated data, as shown below.

```
Odat[1,]
```

```
##   ID       W1        W2 W3 A        Y
## 1  1 3.705826 0.1686546  1 1 7.080206
```

Specifying interventions (`+ action`)

The example below defines two actions on the treatment node. The first action named `"A1"` consists in replacing the distribution of the treatment node `A` with the degenerate distribution at value 1. The second action named `"A0"` consists in replacing the distribution of the treatment node `A` with the degenerate distribution at value 0. As shown below, these interventions are defined by invoking the `+ action` syntax on the existing `DAG` object. This syntax automatically adds and saves the new intervention object within the original `DAG` object, without overwriting it.

```
A1 <- node("A", distr = "rbern", prob = 1)
Dset3 <- Dset3 + action("A1", nodes = A1)
A0 <- node("A", distr = "rbern", prob = 0)
Dset3 <- Dset3 + action("A0", nodes = A0)
```

The added actions can be examined by looking at the result of the call `A(Dset)`. Note that `A(Dset)` returns a list of `DAG.action` objects, with each `DAG.action` encoding a particular post-intervention distribution, i.e., it is a modified copy of the original `DAG` object, where the original distribution of the node `A` is replaced with the degenerate distribution at value 0 or 1, for actions `"A0"` and `"A1"`, respectively.

```
names(A(Dset3))
class(A(Dset3)[["A0"]])
```

Simulating counterfactual data (`sim`)

Simulating counterfactual data is accomplished by calling the function `sim` and specifying its arguments `DAG`, `actions` and `n` to indicate the causal model, interventions, and sample size of interest. Counterfactual data can be simulated for all actions stored in the `DAG` object or a subset by setting the `actions` argument to the vector of the desired action names.

The example below shows how to use the `sim` function to simulate 100,000 observations for each of the two actions, `"A1"` and `"A0"`. These actions were defined as part of the `DAG` object `Dset` above. The call to `sim` below produces a list of two named `data.frame` objects, where each `data.frame` object contains observations simulated from the same post-intervention distribution defined by one particular action only.

```
Xdat1 <- sim(DAG = Dset3, actions = c("A1", "A0"), n = 100000, rndseed = 123)
names(Xdat1)
nrow(Xdat1[["A1"]])
nrow(Xdat1[["A0"]])
```

The format of the output list is easily understood by examining the first row of each `data.frame` object:

```
Xdat1[["A1"]][1, ]
Xdat1[["A0"]][1, ]
```

Defining and evaluating various causal target parameters

Causal parameters defined with `set.targetE`

The first example below defines the causal quantity of interest as the expectation of node `Y` under action `"A1"`, i.e., $E(Y_1)$:

```
Dset3 <- set.targetE(Dset3, outcome = "Y", param = "A1")
```

The true value of the above causal parameter is now evaluated by calling the function `eval.target` and passing the previously simulated counterfactual data object `Xdat1`.

```
eval.target(Dset3, data = Xdat1)$res
```

Alternatively, `eval.target` can be called without the simulated counterfactual data, specifying the sample size argument `n` instead. In this case a counterfactual dataset with the user-specified sample size is simulated first.

```
eval.target(Dset3, n = 100000, rndseed = 123)$res
```

The example below defines the causal target parameter as the ATE on the additive scale, i.e., the expectation of `Y` under action `"A1"` minus its expectation under action `"A0"`, given by $E(Y_1 - Y_0)$:

```
Dset3 <- set.targetE(Dset3, outcome = "Y", param = "A1-A0")
eval.target(Dset3, data = Xdat1)$res
```

```
##   Diff_Y
## 1.281203
```

Similarly, the ATE on the multiplicative scale given by $E(Y_1)/E(Y_0)$ can be evaluated as follows:

```
Dset3 <- set.targetE(Dset3, outcome = "Y", param = "A1/A0")
eval.target(Dset3, data = Xdat1)$res
```

Causal parameters defined with `set.targetMSM`

To specify MSM target causal parameter, the user must provide the following arguments to `set.targetMSM`: (1) the `DAG` object that contains all and only the actions of interest; (2) `outcome`, the name of the outcome node (possibly time-varying); (3) for a time-varying outcome node, the vector of time points `t` that index the outcome measurements of interest; (4) `form`, the regression formula defining the working MSM; (5) `family`, the working model family that is passed on to `glm`, e.g., `family = "binomial"` or `family = "gaussian"` for a logistic or a linear working model; and (6) for time-to-event outcomes, the logical flag `hazard` that indicates whether the working MSM describes discrete-time hazards (`hazard = TRUE`) or survival probabilities (`hazard = FALSE`).

In the examples above, the two actions `"A1"` and `"A0"` are defined as deterministic static interventions on the node `A`, setting it to either constant 0 or 1. Thus, each of these two interventions is uniquely indexed by the post-intervention value of the node `A` itself. In the following example, we instead introduce the variable $d \in \{0,1\}$ to explicitly index each of the two post-intervention distributions when defining the two actions of interest. We then define the target causal parameter as the coefficients of the following linear marginal structural model $m(d \mid \alpha) = \alpha_0 + \alpha_1 d$. As expected, the estimated true value for α_1 obtained below corresponds exactly with the estimated value for the ATE on the additive scale obtained above by running `set.targetE` with the parameter `param = "A1-A0"`.

As just described, we now redefine the actions `"A1"` and `"A0"` by indexing the intervention node formula (the distributional parameter `prob`) with parameter `d` before setting its values to 0 or 1 by introducing an additional new argument named `d` into the `action` function call. This creates an action- specific attribute variable `d` whose value uniquely identifies each of the two actions and that will be included as an additional column variable to the simulating counterfactual data sets.

```
newA <- node("A", distr = "rbern", prob = d)
Dset3 <- Dset3 + action("A1", nodes = newA, d = 1)
Dset3 <- Dset3 + action("A0", nodes = newA, d = 0)
```

Creating such an action-specific attribute `d` allows it to be referenced in the MSM regression formula as shown below:

```
msm.form <- "Y ~ d"
Dset3 <- set.targetMSM(Dset3, outcome = "Y", form = msm.form,
                       family = "gaussian")
msm.res <- eval.target(Dset3, n = 100000, rndseed = 123)
msm.res$coef
```

```
## (Intercept)           d
##    7.385276    1.281203
```

Defining node distributions

To facilitate the comprehension of this subsection, we note that, in the **simcausal** package, simulation of observed or counterfactual data follows the temporal ordering of the nodes that define the DAG object and is *vectorized*. More specifically, the simulation of a dataset with sample size `n` is carried out by first sampling the vector of all `n` observations of the first node, before sampling the vector of all `n` observations of the second node and so on, where the node ranking is defined by the temporal ordering that was explicitly or implicitly specified by the user during the creation of the DAG object (see Section 2.2 for a discussion of temporal ordering).

The distribution of a particular node is specified by passing the name of an evaluable R function to the `distr` argument of the function `node`. Such a distribution function must implement the mapping of `n` independent realizations of the parent nodes into `n` independent realizations of this node. In general, any node with a lower temporal ordering can be defined as a parent. Thus, such a distribution function requires an argument `n`, but will also typically rely on additional input arguments referred to as distributional parameters. In addition, the output of the distribution function must also be a vector of length `n`. Distributional parameters must be either scalars or vectors of `n` realizations of summary measures of the parent nodes. The latter types of distributional arguments are referred to as the *node formula(s)* because they are specified by evaluable R expressions. Distributional parameters are passed as named arguments to the `node` function so they can be mapped uniquely to the relevant argument of the function that is user-specified by the `distr` argument of the `node` function call. The node formula(s) of any given node may invoke the name(s) of any other node(s) with a lower temporal order value. The parents of a particular node are thus defined as the collection of nodes that are referenced by its node formula(s). Note that unlike the values of distributional parameters, the value of the argument `n` of the `distr` function is internally determined during data simulation and is set to the sample size value passed to the `sim` function by the user.

For example, as shown below, the pre-written wrapper function for the Bernoulli distribution `rbern` is defined with two arguments, `n` and `prob`. When defining a node with the `distr` argument set to `"rbern"`, only the second argument must be explicitly user-specified by a distributional parameter named `prob` in the call to the `node` function, e.g., `node("N1", distr="rbern", prob = 0.5)`. The argument `prob` can be either a numeric constant as in the previous example or an evaluable R expression. When `prob` is a numeric constant, the distribution function `rbern` returns `n` iid realizations of the Bernoulli random variable with probability `prob`. When `prob` is an R expression (e.g., see the definition of node `W3` in Section 2.3) that involves parent nodes, the `prob` argument passed to the `rbern` function becomes a vector of length `n`. The value of each of its component is determined by the R expression evaluated using one of the `n` iid realizations of the parent nodes simulated previously. Thus, the resulting simulated independent observations of the child node (e.g., `W3` in Section 2.3) are not necessarily identically distributed if the vector `prob` contains distinct values. We note that the R expression in the `prob` argument is evaluated in the environment

containing the simulated observations of all previous nodes (i.e., nodes with a lower temporal order value).

To see the names of all pre-written distribution wrapper functions that are specifically optimized for use as `distr` functions in the **simcausal** package, invoke `distr.list()`, as shown below:

```
distr.list()
```

```
## [1] "rbern"           "rcat.b0"         "rcat.b1"         "rcat.factor"
## [5] "rcategor"        "rcategor.int"    "rconst"          "rdistr.template"
```

For a template on how to write a custom distribution function, see the documentation `?rdistr.template` and `rdistr.template`, as well as any of the pre-written distribution functions above. For example, the `rbern` function below simply wraps around the standard R function `rbinom` to define the Bernoulli random variable generator:

```
rbern
```

```
## function (n, prob)
## ##     rbinom(n = n, prob = prob, size = 1)##
## <environment: namespace:simcausal>
```

Another example on how to write a custom distribution function to define a custom left-truncated normal distribution function based on the standard R function `rnorm` with arguments `mean` and `sd` is demonstrated below. The truncation level is specified by an additional distributional parameter `minval`, with default value set to 0.

```
rnorm_trunc <- function(n, mean, sd, minval = 0) {
  out <- rnorm(n = n, mean = mean, sd = sd)
  minval <- minval[1]
  out[out < minval] <- minval
  out
}
```

The example below makes use of this function to define the outcome node `Y` with positive values only:

```
Dmin0 <- DAG.empty()
Dmin0 <- Dmin0 +
  node("W", distr = "rbern",
    prob = plogis(-0.5)) +
  node("A", distr = "rbern",
    prob = plogis(-0.5 - 0.3 * W)) +
```

```
  node("Y", distr = "rnorm_trunc",
    mean = -0.1 + 1.2 * A + 0.3 * W,
    sd = 10)
Dmin0set <- set.DAG(Dmin0)
```

In the next example, we overwrite the previous definition of node `Y` to demonstrate how alternative values for the truncation parameter `minval` may be passed by the user as part of the `node` function call:

```
Dmin0 <- Dmin0 +
  node("Y", distr = "rnorm_trunc",
    mean = -0.1 + 1.2 * A + 0.3 * W,
    sd = 10,
    minval = 10)
Dmin10set <- set.DAG(Dmin0)
```

Finally, we illustrate how the `minval` argument can also be defined as a function of parent node realizations:

```
Dmin0 <- Dmin0 +
  node("Y", distr = "rnorm_trunc",
    mean = -0.1 + 1.2 * A + 0.3 * W,
    sd = 10,
    minval = ifelse(A == 0, 5, 10))
Dminset <- set.DAG(Dmin0)
```

As just described, the distributional parameters defining a particular node distribution can be evaluable R expressions, referred to as *node formulas*. These expressions can contain any built-in or user-defined R functions. By default, any user-defined function inside such an R expression is assumed non-vectorized, except for functions on the **simcausal** built-in list of known vectorized functions (this list can be printed by calling `vecfun.all.print()`). We note that the simulation time can often be significantly improved by using vectorized user-defined node formula functions. For example, to register a new user-defined vectorized function `"funname"`, which is not part of the built-in vectorized function list, the user may call `vecfun.add("funname")`. We refer to the package vignette [115] for additional details and examples on how to write custom vectorized node formula functions. We also refer to the same vignette for a simulation demonstrating the performance gains as a result of vectorization.

2.4 Simulation study with multiple time point interventions

In this example we replicate results from the longitudinal data simulation protocol used in two published manuscripts [78, 79]. We first describe the structural equation model that implies the data generating distribution of the observed data, with time-to-event outcome, as reported in Section 5.1 of [79]. We then show how to specify this model using the **simcausal** R interface, simulate observed data, define static and dynamic intervention, simulate counterfactual data, and calculate various causal parameters based on these interventions. In particular, we replicate estimates of true counterfactual risk differences under the dynamic interventions reported in [78], as shown in Section 2.4 (Example 1 for `set.targetE` and Example 1 for `set.targetMSM`).

Specifying the structural equation model

In this section, we demonstrate how to specify the structural equation model described by the following longitudinal data simulation protocol (Section 5.1 of [79]):

1. $L_2(0) \sim \mathcal{B}(0.05)$ where $\mathcal{B}$ denotes the Bernoulli distribution (e.g., $L_2(0)$ represents a baseline value of a time-dependent variable such as low versus high A1c)

2. If $L_2(0) = 1$ then $L_1(0) \sim \mathcal{B}(0.5)$, else $L_1(0) \sim \mathcal{B}(0.1)$ (e.g., $L_1(0)$ represents a time-independent variable such as history of cardiovascular disease at baseline)

3. If $(L_1(0), L_2(0)) = (1,0)$ then $A_1(0) \sim \mathcal{B}(0.5)$, else if $(L_1(0), L_2(0)) = (0,0)$ then $A_1(0) \sim \mathcal{B}(0.1)$, else if $(L_1(0), L_2(0)) = (1,1)$ then $A_1(0) \sim \mathcal{B}(0.9)$, else if $(L_1(0), L_2(0)) = (0,1)$ then $A_1(0) \sim \mathcal{B}(0.5)$ (e.g., $A_1(0)$ represents the binary exposure to an intensified type 2 diabetes pharmacotherapy)

4. for $t = 1, \ldots, 16$ and as long as $Y(t-1) = 0$ (by convention, $Y(0) = 0$):

   a) $Y(t) \sim \mathcal{B}(\frac{1}{1+\exp(-(-6.5+L_1(0)+4L_2(t-1)+0.05*\sum_{j=0}^{t-1} I(L_2(j)=0)))})$ (e.g., Y(t) represents the indicator of failure such as onset or progression of albuminuria)

   b) If $A_1(t-1) = 1$ then $L_2(t) \sim \mathcal{B}(0.1)$, else if $L_2(t-1) = 1$ then $L_2(t) \sim \mathcal{B}(0.9)$, else $L_2(t) \sim \mathcal{B}(\min(1, 0.1 + t/16))$

   c) If $A_1(t-1) = 1$ then $A_1(t) = 1$, else if $(L_1(0), L_2(t)) = (1,0)$ then $A_1(t) \sim \mathcal{B}(0.3)$, else if $(L_1(0), L_2(t)) = (0,0)$ then $A_1(t) \sim \mathcal{B}(0.1)$, else if $(L_1(0), L_2(t)) = (1,1)$ then $A_1(t) \sim \mathcal{B}(0.7)$, else if $(L_1(0), L_2(t)) = (0,1)$ then $A_1(t) \sim \mathcal{B}(0.5)$.

First, the example below shows how to define the nodes `L2`, `L1` and `A1` at time point `t = 0` as Bernoulli random variables, using the distribution function `"rbern"`:

```
library("simcausal")
D <- DAG.empty()
D <- D +
  node("L2", t = 0, distr = "rbern",
    prob = 0.05) +
  node("L1", t = 0, distr = "rbern",
    prob = ifelse(L2[0] == 1, 0.5, 0.1)) +
  node("A1", t = 0, distr = "rbern",
    prob =
      ifelse(L1[0] == 1 & L2[0] == 0, 0.5,
      ifelse(L1[0] == 0 & L2[0] == 0, 0.1,
      ifelse(L1[0] == 1 & L2[0] == 1, 0.9, 0.5))))
```

Second, the example below shows how one may use the `node` function with node formulas based on the square bracket function `'['` to easily define the time-varying nodes `Y`, `L1` and `A1` simultaneously for all subsequent time points `t` ranging from 1 to 16:

```
t.end <- 16
D <- D +
  node("Y", t = 1:t.end, distr = "rbern",
    prob =
      plogis(-6.5 + L1[0] + 4 * L2[t-1] +
        0.05 * sum(I(L2[0:(t-1)] == rep(0, t)))),
    EFU = TRUE) +
  node("L2", t = 1:t.end, distr = "rbern",
    prob =
      ifelse(A1[t-1] == 1, 0.1,
      ifelse(L2[t-1] == 1, 0.9, min(1, 0.1 + t / 16)))) +
  node("A1", t = 1:t.end, distr = "rbern",
    prob =
      ifelse(A1[t-1] == 1, 1,
      ifelse(L1[0] == 1 & L2[t] == 0, 0.3,
      ifelse(L1[0] == 0 & L2[t] == 0, 0.1,
      ifelse(L1[0] == 1 & L2[t] == 1, 0.7, 0.5)))))
lDAG <- set.DAG(D)
```

Note that the `node` formulas specified with the `prob` argument above use the generic time variable `t` both outside and inside the square-bracket vector syntax. For example, the conditional distribution of the time-varying node `Y` is defined by an R expression that contains the syntax `sum(I(L2[0:(t - 1)] == rep(0, t)))`, which evaluates to different R expressions, as `t` ranges from 0 to 16:

1. `sum(I(L2[0] == 0))`, for `t = 1`; and

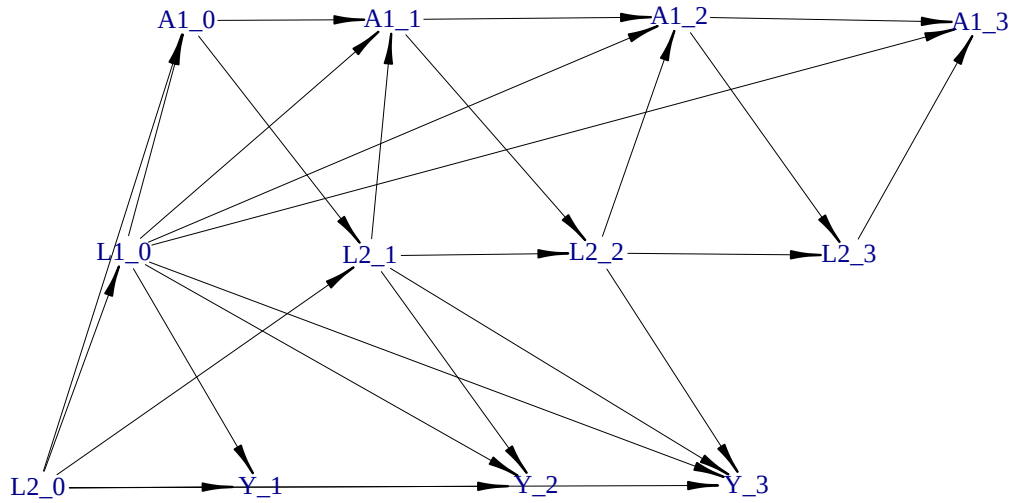

Figure 2.4: Graphical representation of a portion of the structural equation model using a DAG. Only the nodes indexed by time points lower than or equal to 3 are represented.

2. `sum(I(L2[0:1] == c(0, 0)))`, for `t = 2`, ..., `sum(I(L2[0:16] == c(0, ..., 0)))`, for `t = 16`.

For more details on the specification of node formulas, see Section 2.3.

One can visualize the observed data generating distribution defined in the `lDAG` object as shown in Figures 2.4 by calling `plotDAG`. Note that the appearance of the resulting diagram can be customized with additional arguments, as demonstrated in the package vignette [115].

Simulating observed data (`sim`)

Simulating observed data is accomplished by calling the function `sim` and specifying its arguments `DAG` and `n` that indicate the causal model and sample size of interest. Below is an example of how to simulate an observed dataset with $10,000$ observations using the causal model defined previously. The output is a `data.frame` object.

```
Odat <- sim(DAG = lDAG, n = 10000, rndseed = 123)
Odat[1,]
```

Specifying interventions (`+ action`)

Dynamic interventions

The following two dynamic interventions on the time-varying node `A1` of the structural equation model encoded by the previously defined `lDAG` object were studied in [78]: '*Initiate treatment A_1 the first time t that the covariate L_2 is greater than or equal to θ and continue*

treatment thereafter (i.e., $\bar{A}_1(t-1)=0$ *and* $A(t)=1, A(t+1)=1, \ldots$*)'*, for $\theta = 0, 1$. The example below demonstrates how to specify these two dynamic interventions.

First, we define the list of intervention nodes and their post-intervention distributions. Note that these distributions are indexed by the attribute `theta`, whose value is not yet defined:

```
act_theta <-c(
  node("A1", t = 0, distr = "rbern",
    prob = ifelse(L2[0] >= theta , 1, 0)),
  node("A1", t = 1:(t.end), distr = "rbern",
    prob = ifelse(A1[t-1] == 1, 1, ifelse(L2[t] >= theta, 1, 0))))
```

Second, we add the two dynamic interventions to the `lDAG` object while defining the value of `theta` for each intervention:

```
Ddyn <- lDAG
Ddyn <- Ddyn + action("A1_th0", nodes = act_theta, theta = 0)
Ddyn <- Ddyn + action("A1_th1", nodes = act_theta, theta = 1)
```

We refer to the argument `theta` passed to the `+action` function as an *action attribute*.

One can select and inspect particular actions saved in a DAG object by invoking the function `A()`:

```
class(A(Ddyn)[["A1_th0"]])
A(Ddyn)[["A1_th0"]]
```

The distribution of some or all of the the intervention nodes that define an action saved within a `DAG` object can be modified by adding a new intervention object with the same action name to the `DAG` object. The new intervention object can involve actions on only a subset of the original intervention nodes for a partial modification of the original action definition. For example, the code below demonstrates how the existing action `"A1_th0"` with the previously defined dynamic and deterministic intervention on the node `A1[0]` is partially modified by replacing the intervention distribution for the node `A1[0]` with a deterministic and static intervention defined by a degenerate distribution at value 1. Note that the other intervention nodes previously defined as part of the action `"A1_th0"` remain unchanged.

```
A(Ddyn)[["A1_th0"]]$A1_0
Ddyntry <- Ddyn +
  action("A1_th0", nodes = node("A1", t = 0, distr = "rbern", prob = 0))
A(Ddyntry)[["A1_th0"]]$A1_0
```

Similarly, some or all of the action attributes that define an action saved within a `DAG` object can be modified by adding a new intervention object with the same action name but

a different attribute value to the `DAG` object. This functionality is demonstrated with the example below in which the previous value 0 of the action attribute `theta` that defines the action named `"A1_th0"` is replaced with the value 1 and in which a new attribute `newparam` is simultaneously added to the previously defined action `"A1_th0"`:

```
A(Ddyntry)[["A1_th0"]]
Ddyntry <- Ddyntry +
  action("A1_th0", nodes = act_theta, theta = 1, newparam = 100)
A(Ddyntry)[["A1_th0"]]
```

Static interventions

Here we diverge from the replication of simulation results presented in [78]. Instead, we build on the structural equation model introduced in that paper to illustrate the specification of static interventions on the treatment nodes `A1`. These static interventions are defined by more or less early treatment initiation during follow-up followed by subsequent treatment continuation. Each of these static interventions is thus uniquely identified by the time when the measurements of the time-varying node `A1` switch from value 0 to 1. The time of this value switch is represented by the parameter `tswitch` in the code below. Note that the value `tswitch = 16` identifies the static intervention corresponding with no treatment initiation during follow-up in our example while the values 0 through 15 represent 16 distinct interventions representing increasingly delayed treatment initiation during follow-up.

First, we define the list of intervention nodes and their post-intervention distributions. Note that these distributions are indexed by the attribute `tswitch`, whose value is not yet defined:

```
`%+%` <- function(a, b) paste0(a, b)
Dstat <- lDAG
act_A1_tswitch <- node("A1",t = 0:(t.end), distr = "rbern",
                       prob = ifelse(t >= tswitch, 1, 0))
```

Second, we add the 17 static interventions to the `lDAG` object while defining the value of `tswitch` for each intervention:

```
tswitch_vec <- (0:t.end)
for (tswitch_i in tswitch_vec) {
  abar <- rep(0, length(tswitch_vec))
  abar[which(tswitch_vec >= tswitch_i)] <- 1
  Dstat <- Dstat + action("A1_ts"%+%tswitch_i,
                          nodes = act_A1_tswitch,
                          tswitch = tswitch_i,
                          abar = abar)
```

```
}
```

Note that in addition to the action attribute `tswitch`, each intervention is also indexed by an additional action attribute `abar` that also uniquely identifies the intervention and that represents the actual sequence of treatment decisions that defines the intervention, i.e., $\bar{a}(tswitch-1)=0$, $a(tswitch)=1$, ...:

```
A(Dstat)[["A1_ts3"]]
```

The purpose of this additional action attribute `abar` will become clear when we illustrate the definition of target parameters defined by working MSMs based on these 17 static interventions in Section 2.4 (Example 2 of `set.targetMSM`).

Simulating counterfactual data (`sim`)

Simulating counterfactual data is accomplished by calling the function `sim` and specifying its arguments `DAG`, `actions` and `n` to indicate the causal model, interventions, and sample size of interest. The counterfactual data can be simulated for all actions stored in the `DAG` object or a subset by setting the `actions` argument to the vector of the desired-action names.

The example below shows how to use the `sim` function to simulate 200,000 observations for each of the two dynamic actions, `"A1_th0"` and `"A1_th1"`, defined in Section 2.4. The call to `sim` below produces a list of two named `data.frame` objects, where each `data.frame` object contains observations simulated from the same post-intervention distribution defined by one particular action only.

```
Xdyn <- sim(Ddyn, actions = c("A1_th0", "A1_th1"),
            n = 200000, rndseed = 123)
```

The default format of the output list generated by the `sim` function is easily understood by examining the first row of each `data.frame` object:

```
Xdyn[["A1_th0"]][1, ]
Xdyn[["A1_th1"]][1, ]
```

Converting a dataset from *wide* to *long* format (`DF.to.long`)

The specification of structural equation models based on time-varying nodes such as the one described in Section 2.4 allows for simulated (observed or counterfactual) data to be structured in either long or wide formats. Below, we illustrate these two alternatives. We note that, by default, simulated (observed or counterfactual) data from the `sim` function are stored in wide format. The data output format from the `sim` function can, however, be

changed to the long format by setting the `wide` argument of the `sim` function to `FALSE` or, equivalently, by applying the function `DF.to.long` to an existing simulated dataset in wide format.

The following code demonstrates the default data formatting behavior of the `sim` function and how this behavior can be modified to generate data in the long format:

```
Odat.wide <- sim(DAG = lDAG, n = 1000, wide = TRUE, rndseed = 123)
Odat.wide[1:2, 1:16]
```

```
##   ID L2_0 L1_0 A1_0 Y_1 L2_1 A1_1 Y_2 L2_2 A1_2 Y_3 L2_3 A1_3 Y_4 L2_4 A1_4
## 1  1    0    0    0   0    0    0   0    0    0   0    1    0   1   NA   NA
## 2  2    0    0    0   0    0    0   0    0    0   0    0    0   0    0    0
```

```
Odat.long <- sim(DAG = lDAG, n = 1000, wide = FALSE, rndseed = 123)
Odat.long[1:7, ]
```

```
##   ID L1 t L2 A1  Y
## 1  1  0 0  0  0 NA
## 2  1  0 1  0  0  0
## 3  1  0 2  0  0  0
## 4  1  0 3  1  0  0
## 5  1  0 4 NA NA  1
## 6  2  0 0  0  0 NA
## 7  2  0 1  0  0  0
```

Note that the first observation in `Odat.wide` contains `NA`s following `Y_4`. As described in Section 2.2, this is due to the fact that the node `Y` was defined earlier as an end of follow-up (EFU) event (using argument `EFU=TRUE`). That is, `Y_4=1` indicates that the first subject has reached the end of the follow-up at time point $t = 4$ (i.e., was right-censored), therefore, all of the subsequent columns following `Y_4` are replaced with `NA` (missing) value. This is also the reason why we only see 5 rows of data on subject with `ID=1` in the above long format dataset `Odat.long`. Also note that in `Odat.long`, the value of `Y` is always `NA` (missing) for `t=0`, since the node `Y` was only defined for time-points $t > 0$.

Defining and evaluating various causal target parameters

Causal parameters defined with `set.targetE`

Example 1. In the example below, we first define two causal target parameters as two vectors, each containing the expectations of the node `Y[t]`, for time points `t`=1, ..., 16, under the post-intervention distribution defined by one of the two dynamic interventions `"A1_th0"` and `"A1_th1"` defined in Section 2.4. Second, we evaluate these target parameters using the counterfactual data simulated previously in Section 2.4 and we map the resulting estimates of cumulative risks into estimates of survival probabilities. We also plot the corresponding

two counterfactual survival curves using the **simcausal** routine `plotSurvEst` as shown in Figure 2.5. Finally, we note that Figure 2.5 replicates the simulation study results reported in Figure 4 of [78].

```
Ddyn <- set.targetE(Ddyn, outcome = "Y", t = 1:16, param = "A1_th1")
surv_th1 <- 1 - eval.target(Ddyn, data = Xdyn)$res
Ddyn <- set.targetE(Ddyn, outcome = "Y", t = 1:16, param = "A1_th0");
surv_th0 <- 1 - eval.target(Ddyn, data = Xdyn)$res


plotSurvEst(surv = list(d_theta1 = surv_th1, d_theta0 = surv_th0),
            xindx = 1:17,
            ylab = "Counterfactual survival for each intervention",
            ylim = c(0.75,1.0))
```

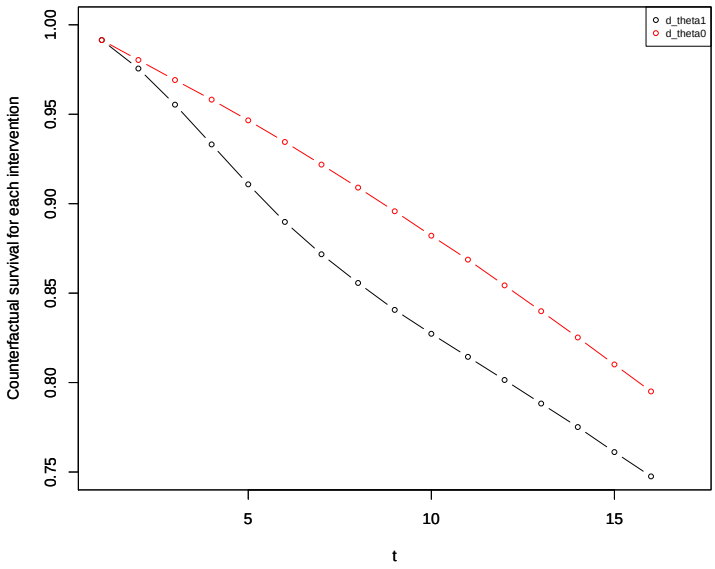

Figure 2.5: Estimates of the true survival curves under the two dynamic interventions.

Example 2. In the example below, we first define the causal target parameter as the ATE on the additive scale (cumulative risk differences) for the two dynamic interventions (`"A1_th1"` and `"A1_th0"`) defined in Section 2.4 at time point `t = 12`. Second, we evaluate this target parameter using the previously simulated counterfactual data from Section 2.4.

ATE on the additive scale:

```
Ddyn <- set.targetE(Ddyn, outcome = "Y", t = 12, param = "A1_th1-A1_th0")
(psi <- round(eval.target(Ddyn, data = Xdyn)$res, 3))

## Diff_Y_12
##     0.053
```

We also note that the above result for the ATE (0.053) replicates the simulation result reported for ψ in Section 5.1 and Figure 4 of [78], where ψ was defined as the difference between the cumulative risks of failure at $t_0 = 12$ for the two dynamic interventions d_1 and d_0.

Causal parameters defined with `set.targetMSM`

In Section 2.3, we described the arguments of the function `set.targetMSM` that the user must specify to define MSM target causal parameters. They include the specification of the argument `form` which encodes the working MSM formula. This formula can only be a function of the time index `t`, action attributes that uniquely identify each intervention of interest, and baseline nodes (defined as nodes that precede the earliest intervention node). Both baseline nodes that are measurements of time-varying nodes and time-varying action attributes must be referenced in the R expression passed to the `form` argument within the wrapping syntax `S(...)` as illustrated in several examples below.

Example 1. Working dynamic MSM for survival probabilities over time. Here, we illustrate the evaluation of the counterfactual survival curves $E(Y_{d_\theta}(t))$ for $t = 1, \ldots, 16$ under the dynamic interventions d_θ for $\theta = 0, 1$ introduced in Section 2.4 using the following pooled working logistic MSM (MSM 1):

$$\text{expit}\left(\alpha_0 + \alpha_1\theta + \alpha_2 t + \alpha_3 t\theta\right),$$

where the true values of the coefficients $(\alpha_i, i = 0, \ldots, 3)$ define the target parameters of interest. First, we define these target parameters:

```
msm.form <- "Y ~ theta + t + I(theta*t)"
Ddyn <- set.targetMSM(Ddyn, outcome = "Y", t = 1:16, form = msm.form,
                      family = "binomial", hazard = FALSE)
```

Note that when the outcome is a time-varying node of type `EFU`, the argument `hazard = FALSE` indicates that the working MSM of interest is a model for survival probabilities. The argument `family = "binomial"` indicates that the working model is a logistic model. Second, we evaluate the coefficients of the working model:

```
MSMres1 <- eval.target(Ddyn, n = 10000, rndseed = 123)
MSMres1$coef
```

We also note that no previously simulated counterfactual data were passed as argument to the function `eval.target` above. Instead, the sample size argument `n` was specified and the routine will thus first sample `n = 10,000` observations from each of the two post-intervention distributions before fitting the working MSM with these counterfactual data to derive estimates of the true coefficient values. Alternatively, the user could have passed the previously simulated counterfactual data. Note however that in this case, the user must either simulate counterfactual data by calling the `sim` function with the argument `LTCF = "Y"` or convert the previously simulated counterfactual data with the *last time point value carried forward* imputation function `doLTCF`. Both approaches are described in the **simcausal** package vignette Section 4.7 [115].

The resulting coefficient estimates for MSM 1 can be mapped into estimates of the two counterfactual survival curves and plotted as shown on the left in Figure 2.6 using the **simcausal** `plotSurvEst` function.

Next, we modify the previous working model formula by specifying a saturated MSM to directly replicate the results reported in Figure 4 of [78] that are based on a non-parametric MSM approach (MSM 2):

```
msm.form <- "Y ~ theta + as.factor(t) + as.factor(t):theta "
Ddyn <- set.targetMSM(Ddyn, outcome = "Y", t = 1:16, formula = msm.form,
                      family = "binomial", hazard = FALSE)
MSMres2 <- eval.target(Ddyn, n = 200000, rndseed = 123)
MSMres2$coef
```

Finally, we plot the resulting survival curves obtained from MSM 2 as shown on the right in Figure 2.6. The resulting estimates of the survival curves replicate those reported in Figure 4 of [78].

Example 2. Working static MSM for discrete-time hazards over time. Here, we illustrate the evaluation of discrete-time hazards $E(Y_{\bar{a}}(t))|Y_{\bar{a}}(t-1)=0)$, for $t=1,...,16$ under the 17 static interventions introduced in Section 2.4 using the following pooled working logistic MSM:

$$expit\Big(\alpha_0+\alpha_1 t+\alpha_2\frac{1}{t}\sum_{j=0}^{t-1}a(j)+\alpha_3\sum_{j=0}^{t-1}a(j)\Big),$$

where we use the notation $\bar{a}=(a(0),a(1),...,a(16))$ to denote the 17 static intervention regimens on the time-varying treatment node `A1`. Note that the time-varying action attribute `abar` introduced in Section 2.4 directly encodes the 17 treatment regimens values $\bar{a}$ referenced in the MSM working model above. To evaluate the target parameters α_j above, for $j=0,...,3$, we first simulate counterfactual data for the 17 static interventions of interest as follows:

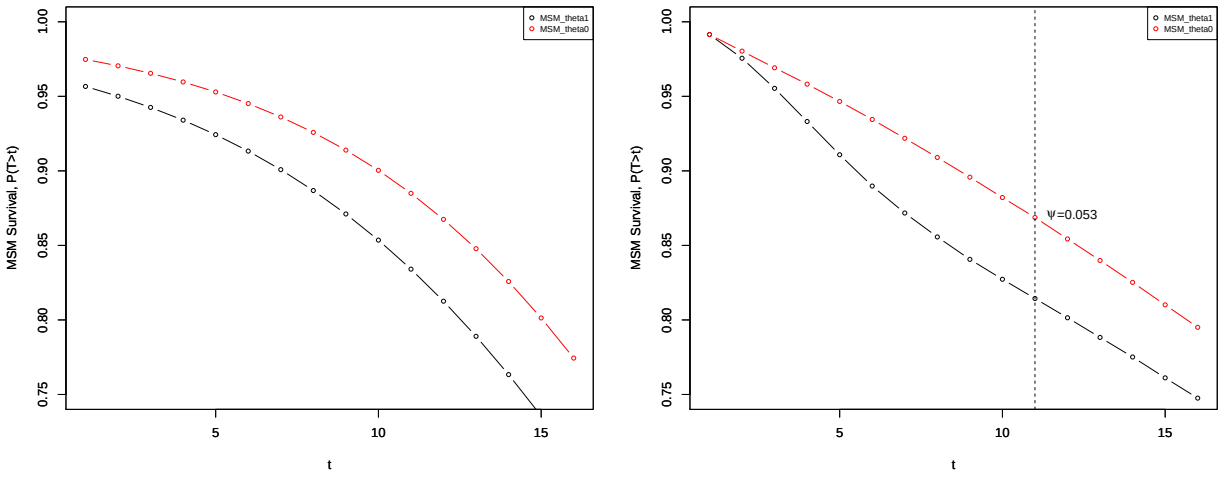

Figure 2.6: Survival curve estimates evaluated based on working MSM 1 (left) and saturated MSM 2 (right).

```
Xts <- sim(Dstat, actions = names(A(Dstat)), n = 1000, rndseed = 123)
```

Second, we define the target parameters and estimate them using the counterfactual data just simulated as follows:

```
msm.form_1 <- "Y ~  t +
               S(mean(abar[0:(t-1)])) + I(t*S(mean(abar[0:(t-1)])))"
Dstat <- set.targetMSM(Dstat, outcome = "Y", t = 1:16, form = msm.form_1,
                       family = "binomial", hazard = TRUE)
MSMres <- eval.target(Dstat, data = Xts)
MSMres$coef
```

Note that the working MSM formulas can reference arbitrary summary measures (functions) of time-varying action attributes such as `abar`. The square-bracket `'['` syntax can then be used to identify specific elements of the time-varying action attributes in the same way it can be used in node formulas to reference particular measurements of time-varying nodes. For example, the term `sum(abar[0:t])` indicates a summation over the elements of the action attribute `abar` indexed by time points lower than or equal to value `t` and the syntax `S(abar[max(0, t - 2)])` creates a summary measure representing time-lagged values of `abar` that are equal to `abar[0]` if `t`< 2 and to `abar[t-2]` if `t`≥ 2. Note also that references to time-varying action attributes in the working MSM formula must be wrapped within a call to the `S(...)` function, e.g., `Y`$\sim$`t + S(mean(abar[0:t]))`.

The `eval.target` function returns a list with the following named attributes: the working MSM fit returned by a `glm` function call (`msm`), the coefficient estimates (`coef`), the mapping

(`S.msm.map`) of the formula terms defined by expressions enclosed within the `S(...)` function into the corresponding variable names in the design matrix that was used to implement the regression, and the design matrix (`df_long`) stored as a list of `data.table` objects from the R package **data.table** [33]. Each of these `data.table` objects contains counterfactual data indexed by a particular intervention. These counterfactual data are stored in long format with possibly additional new columns representing terms in the working MSM formula defined by expressions enclosed with the `S()` function. The design matrix can be derived by row binding these `data.table` objects.

```
names(MSMres)
MSMres$S.msm.map
names(MSMres$df_long)
MSMres$df_long[["A1_ts2"]]
```

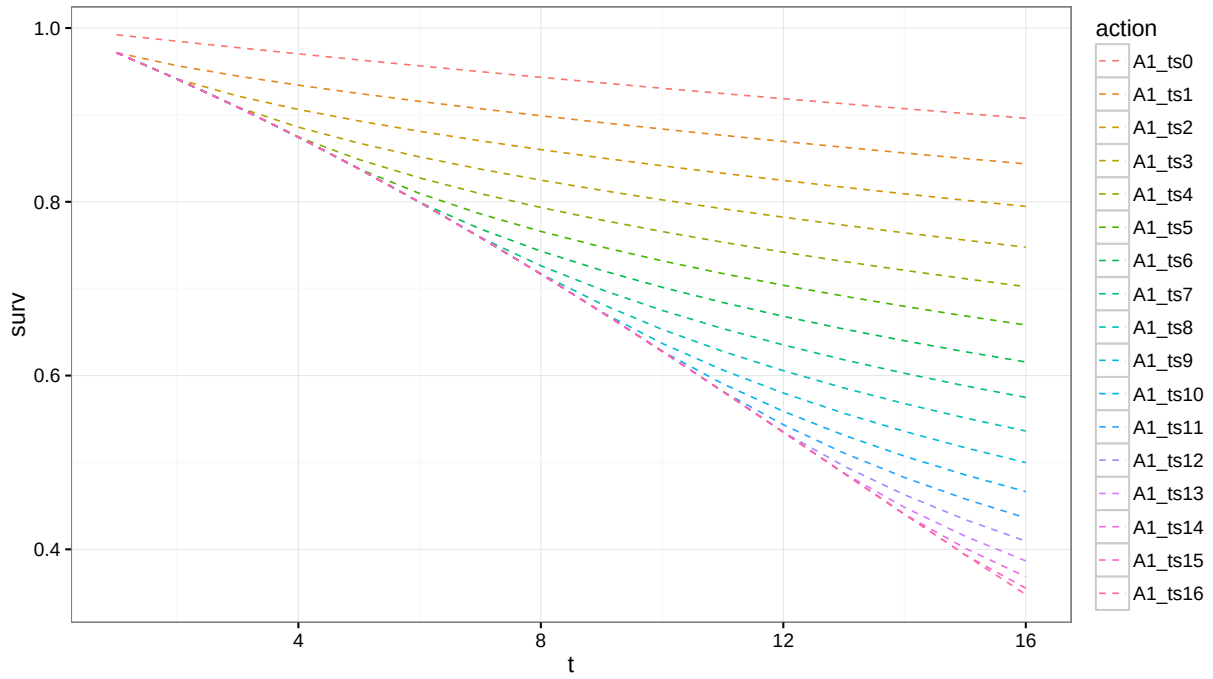

Figure 2.7: Survival curve estimates evaluated based on working MSM 2.

Finally, we plot the resulting counterfactual survival curve estimates using the function `survbyMSMterm` (source code provided in a supplementary R script), as shown in Figure 2.7:

```
survMSMh_wS <- survbyMSMterm(MSMres = MSMres, t_vec = 1:16,
                             MSMtermName = "mean(abar[0:(t - 1)])")
print(plotsurvbyMSMterm(survMSMh_wS))
```

Covariates in $P(A\|L)$	N	A(0) Bias*10	A(0) MSE*10	A(1) Bias*10	A(1) MSE*10
Confounder(s) only	300	0.576	1.803	0.752	1.690
	1000	0.278	0.725	0.374	0.688
	10000	0.062	0.139	0.069	0.147
Confounder(s) &	300	0.572	1.714	0.785	1.489
risk factors	1000	0.250	0.764	0.304	0.665
	10000	0.071	0.121	0.077	0.120

Table 2.1: Replication of the simulation results from [68] for Scenario 1.

Additional examples of working MSMs are available in the package vignette [115], which includes the examples of dynamic MSM for discrete-time hazards and dynamic MSMs that evaluate effect modification by a baseline covariate.

2.5 Replication study of the impact of misspecification of propensity score models

Replication study results

In this section, we use the **simcausal** package for replicating a simulation study from [68]. Specifically, we replicate the results reported in Tables II and IV of that paper. We first specify the observed data generating distribution using the two structural equation models corresponding with Scenarios 1 and 3 described in [68]. Second, for each scenario, we evaluate the true values of the coefficients of the MSM using counterfactual data and compare them to those reported by [68]. Finally, for each scenario, we implement the same inverse probability weighting (IPW) estimators of these MSM coefficients and evaluate their performances using the same two metrics (bias and mean squared error) as in [68]. We refer to next Section 2.5 for the description of the details on how the **simcausal** package was used to conduct this replication study. The R code that fully reproduces the tables presented in this section is provided as a supplementary R script.

Our replication results for Scenarios 1 and 3 are reported in Table 2.1 and Table 2.3, respectively. The simulation results, as they were originally reported by [68], are presented in Table 2.2 and Table 2.4. We note that our results closely match those originally reported in [68].

Covariates in $P(A\|L)$	N	A(0) Bias*10	A(0) MSE*10	A(1) Bias*10	A(1) MSE*10
Lefebvre et al.: Confounder(s) only	300	0.768	1.761	0.889	1.728
	1000	0.265	0.761	0.312	0.723
	10000	0.057	0.146	0.086	0.120
Lefebvre et al.: Confounder(s) &	300	0.757	1.642	0.836	1.505
risk factors	1000	0.283	0.718	0.330	0.638
	10000	0.056	0.139	0.081	0.114

Table 2.2: Simulation results for Scenario 1 as reported in Table II of [68].

Covariates in $P(A\|L)$	N	A(0) Bias*10	A(0) MSE*10	A(1) Bias*10	A(1) MSE*10
Confounder(s) only	300	-0.179	1.238	0.157	1.102
	1000	-0.341	0.413	-0.137	0.363
	10000	-0.347	0.054	-0.177	0.046
Confounder(s) &	300	-0.151	1.156	0.110	0.890
risk factors	1000	-0.266	0.348	-0.093	0.271
	10000	-0.354	0.050	-0.190	0.034
Confounder(s) &	300	1.397	3.966	2.014	3.854
IVs	1000	0.919	2.016	1.200	1.989
	10000	0.438	0.605	0.457	0.595
Confounder(s),	300	1.304	4.010	1.966	3.841
IVs & risk factors	1000	0.936	2.082	1.208	2.027
	10000	0.375	0.644	0.422	0.626
Mis-specified	300	2.742	3.203	5.542	5.437
	1000	2.598	1.737	5.188	3.739
	10000	2.407	0.809	5.009	2.730
Full Model	300	1.383	4.028	2.109	3.924
	1000	0.934	2.020	1.285	1.926
	10000	0.417	0.607	0.435	0.609

Table 2.3: Replication of the simulation results from [68] for Scenario 3.

Covariates in $P(A\|L)$	N	A(0) Bias*10	A(0) MSE*10	A(1) Bias*10	A(1) MSE*10
Lefebvre et al.: Confounder(s) only	300	-0.080	1.170	0.099	1.155
	1000	-0.371	0.385	-0.035	0.331
	10000	-0.368	0.056	-0.203	0.043
Lefebvre et al.: Confounder(s) &	300	-0.110	1.092	0.112	0.865
risk factors	1000	-0.330	0.340	-0.108	0.245
	10000	-0.378	0.051	-0.207	0.037
Lefebvre et al.: Confounder(s) &	300	1.611	3.538	2.069	3.841
IVs	1000	0.824	2.063	1.245	2.188
	10000	0.241	0.684	0.379	0.622
Lefebvre et al.: Confounder(s),	300	1.600	3.477	2.143	3.598
IVs & risk factors	1000	0.867	2.053	1.170	2.043
	10000	0.235	0.676	0.372	0.625
Lefebvre et al.: Mis-specified	300	3.146	3.326	5.591	5.494
	1000	2.460	1.700	5.258	3.851
	10000	2.364	0.832	4.943	2.705
Lefebvre et al.: Full Model	300	1.524	3.648	2.221	3.907
	1000	0.878	2.099	1.185	2.099
	10000	0.240	0.679	0.377	0.630

Table 2.4: Simulation results for Scenario 3 as reported in Table IV of [68].

Additional details and replication code

A number of IPW estimators were considered in this simulation study, each estimator defined by a distinct model for the propensity scores $P(A(0)|L(0))$ and $P(A(1)|A(0), L(1))$. To estimate these propensity scores we used the same models presented in Table I of [68] for Scenarios 1 and 3. We considered three sample sizes $N = 300$; $1,000$; and $10,000$, and we report the bias of each IPW estimator, multiplied by 10 (*Bias*10*) and the mean-squared error, also multiplied by 10 (*MSE*10*) in Tables 2.1 and 2.3.

Replicating Scenario 1

To carry out the simulation study, we first define a new distribution function `rbivNorm` for simulating observations from a bivariate normal distribution with a user-specified mean vector (specified by the argument `mu`) and a user-specified covariance matrix (specified by the arguments `var1`, `var2`, and `rho` to represent the diagonal and off-diagonal scalars, respectively). This new distribution function is based on Cholesky decomposition of the covariance matrix and independent observations simulated from the standard normal distribution which are provided by the input argument `norms`. The argument `whichbiv` indicates whether the

function should return independent observations from the first or second element of the bivariate normal vector.

```
> rbivNorm <- function(n, whichbiv, norms, mu, var1 = 1, var2 = 1, rho = 0.7) {
+   whichbiv <- whichbiv[1]; var1 <- var1[1]; var2 <- var2[1]; rho <- rho[1]
+   sigma <- matrix(c(var1, rho, rho, var2), nrow = 2)
+   Scol <- chol(sigma)[, whichbiv]
+   bivX <- (Scol[1] * norms[, 1] + Scol[2] * norms[, 2]) + mu
+   bivX
+ }
```

Second, using this distribution function, we define the structural equation model specified for data simulation according to Scenario 1 in [68].

```
> `%+%` <- function(a, b) paste0(a, b)
> Lnames <- c("LO1", "LO2", "LO3", "LC1")
> D <- DAG.empty()
> for (Lname in Lnames) {
+   D <- D +
+     node(Lname%+%".norm1", distr = "rnorm", mean = 0, sd = 1) +
+     node(Lname%+%".norm2", distr = "rnorm", mean = 0, sd = 1)
+ }
> D <- D +
+   node("LO1", t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+     norms = c(LO1.norm1, LO1.norm2),
+     mu = 0) +
+   node("LO2", t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+     norms = c(LO2.norm1, LO2.norm2),
+     mu = 0) +
+   node("LO3", t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+     norms = c(LO3.norm1, LO3.norm2),
+     mu = 0) +
+   node("LC1", t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+     norms = c(LC1.norm1, LC1.norm2),
+     mu = {if (t == 0) {0} else {-0.30 * A[t-1]}}) +
+   node("alpha", t = 0:1, distr = "rconst",
+     const = {if(t == 0) {log(0.6)} else {log(1.0)}}) +
+   node("A", t = 0:1, distr = "rbern",
+     prob =
+       plogis(alpha[t] +
+       log(5)*LC1[t] + {if(t == 0) {0} else {log(5)*A[t-1]}})) +
+   node("Y", t = 1, distr = "rnorm",
+     mean = (0.98 * LO1[t] + 0.58 * LO2[t] + 0.33 * LO3[t] +
+             0.98 * LC1[t] - 0.37 * A[t]),
```

```
+     sd = 1)
> DAGO.sc1 <- set.DAG(D)
```

Third, we define the target parameter as the coefficients β_1 and β_2 of the following correctly specified marginal structural model:

$$E[Y_{a(0),a(1)}] = \beta_0 + \beta_1 a(0) + \beta_2 a(1),$$

defined by the following four possible static and deterministic interventions $(a(0), a(1))$ on the treatment process $(A(0), A(1))$: $(0,0)$, $(1,0)$, $(0,1)$, and $(1,1)$.

```
> defAct <- function (Dact) {
+   act.At <- node("A", t = 0:1, distr = "rbern", prob = abar[t])
+   Dact <- Dact +
+     action("A00", nodes = act.At, abar = c(0, 0)) +
+     action("A10", nodes = act.At, abar = c(1, 0)) +
+     action("A01", nodes = act.At, abar = c(0, 1)) +
+     action("A11", nodes = act.At, abar = c(1, 1))
+   return(Dact)
+ }
> Dact.sc1 <- defAct(DAGO.sc1)
> msm.form <- "Y ~ S(abar[0]) + S(abar[1])"
> Dact.sc1 <- set.targetMSM(Dact.sc1, outcome = "Y", t = 1,
+                           form = msm.form, family = "gaussian")
```

Fourth, we evaluate the true values of these MSM coefficients using the `eval.target` function and note that our results closely match the true value of the MSM coefficients reported in Table II of [68]:

```
> repstudy2.sc1.truetarget <- function() {
+   trueMSMreps.sc1 <- NULL
+   reptrue <- 50
+   for (i in (1:reptrue)) {
+     res.sc1.i <- eval.target(Dact.sc1, n = 500000)$coef
+     trueMSMreps.sc1 <- rbind(trueMSMreps.sc1, res.sc1.i)
+   }
+   return(trueMSMreps.sc1)
+ }
> f1name <- "replication_dat/trueMSMreps.sc1.Rdata"
> if (file.exists(f1name)) {
+   load(f1name)
+ } else {
+   trueMSMreps.sc1 <- repstudy2.sc3.truetarget()
+   save(list = "trueMSMreps.sc1", file = f1name)
```

```
+ }
> trueMSM.sc1 <- apply(trueMSMreps.sc1, 2, mean)
> print(trueMSM.sc1)

##   (Intercept)    S(abar[0])    S(abar[1])
##  0.0001540635 -0.2941187264 -0.3700397969
```

Note that the true values of the MSM coefficients above were obtained from the averages of coefficient estimates obtained from several simulated counterfactual data sets. This approach was implemented to avoid the memory limitation that can be encountered when trying to simulate a single very large counterfactual data set. Finally, using the R code provided as a supplementary script file, we replicate the IPW estimation results for Scenario 1 as presented originally in Table II of [68].

Replicating Scenario 3

Next, using the same approach described above, we replicate the simulation results for Scenario 3 reported in Table IV of [68]. We start by defining the structural equation model specified for data simulation according to Scenario 3 in [68] as follows:

```
> `%+%` <- function(a, b) paste0(a, b)
> Lnames <- c("LO1", "LO2", "LO3", "LE1", "LE2", "LE3", "LC1", "LC2", "LC3")
> D <- DAG.empty()
> for (Lname in Lnames) {
+   D <- D +
+     node(Lname%+%".norm1", distr = "rnorm") +
+     node(Lname%+%".norm2", distr = "rnorm")
+ }
> coefAi <- c(-0.10, -0.20, -0.30)
> sdLNi <- c(sqrt(1), sqrt(5), sqrt(10))
> for (i in (1:3)) {
+   D <- D +
+     node("LO"%+%i, t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+       mu = 0,
+       params = list(norms = "c(LO"%+%i%+%".norm1, LO"%+%i%+%".norm2)")) +
+     node("LE"%+%i, t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+       mu = 0, var1 = 1, var2 = 1, rho = 0.7,
+       params = list(norms = "c(LE"%+%i%+%".norm1, LE"%+%i%+%".norm2)")) +
+     node("LC"%+%i, t = 0:1, distr = "rbivNorm", whichbiv = t + 1,
+       mu = {if (t == 0) {0} else {.(coefAi[i]) * A[t-1]}},
+       params = list(norms = "c(LC"%+%i%+%".norm1, LC"%+%i%+%".norm2)")) +
+     node("LN"%+%i, t = 0:1, distr = "rnorm",
+       mean = 0, sd = .(sdLNi[i]))
```

```
+ }
> D <- D +
+   node("alpha", t = 0:1, distr = "rconst",
+     const = {if(t == 0) {log(0.6)} else {log(1.0)}}) +
+   node("A", t = 0:1, distr = "rbern",
+     prob = plogis(alpha[t] +
+                   log(5) * LC1[t] + log(2) * LC2[t] + log(1.5) * LC3[t] +
+                   log(5) * LE1[t] + log(2) * LE2[t] + log(1.5) * LE3[t] +
+                   {if (t == 0) {0} else {log(5) * A[t-1]}})) +
+   node("Y", t = 1, distr = "rnorm",
+     mean = 0.98 * LO1[t] + 0.58 * LO2[t] + 0.33 * LO3[t] +
+            0.98 * LC1[t] + 0.58 * LC2[t] + 0.33 * LC3[t] - 0.39 * A[t],
+     sd = 1)
> DAGO.sc3 <- set.DAG(D)
```

Similar to Scenario 1, we then define the same four actions on the new `DAG` object before defining and evaluating the causal target parameter of interest. We note that our results match the true value of the MSM coefficients reported in Table IV of [68]. Finally, using the R code provided as a supplementary script file, we replicate the IPW estimation results for Scenario 3 as presented originally in Table IV of [68].

```
> Dact.sc3 <- defAct(DAGO.sc3)
> msm.form <- "Y ~ S(abar[0]) + S(abar[1])"
> Dact.sc3 <- set.targetMSM(Dact.sc3, outcome = "Y", t = 1,
+                           form = msm.form, family = "gaussian")
> repstudy2.sc3.truetarget <- function() {
+   trueMSMreps.sc3 <- NULL
+   reptrue <- 50
+   for (i in (1:reptrue)) {
+     res.sc3.i <- eval.target(Dact.sc3, n = 500000)$coef
+     trueMSMreps.sc3 <- rbind(trueMSMreps.sc3, res.sc3.i)
+   }
+   return(trueMSMreps.sc3)
+ }
> f2name <- "replication_dat/trueMSMreps.sc3.Rdata"
> if (file.exists(f2name)) {
+   load(f2name)
+ } else {
+   trueMSMreps.sc3 <- repstudy2.sc3.truetarget()
+   save(list = "trueMSMreps.sc3", file = f2name)
+ }
> trueMSM.sc3 <- apply(trueMSMreps.sc3, 2, mean)
> print(trueMSM.sc3)
```

```
##   (Intercept)    S(abar[0])    S(abar[1])
##  0.0001690372 -0.3134283871 -0.3901595327
```

2.6 Discussion

In this chapter we described how our simulation package can be used for creating a wide range of artificial datasets often encountered in medical and public health applications of causal inference methods. Specifically, we demonstrated that the **simcausal** R package is a flexible tool that facilitates the conduct of transparent and reproducible simulation studies. The package allows the user to simulate complex longitudinal data structures based on structural equation models using a novel interface which allows concise and intuitive expression of complex functional dependencies for a large number of nodes. We also argued that such complex simulations are often necessary when one tries to conduct a realistic simulation study that attempts to replicate a large variety of scenarios one might expect to see from a true data-generating process. The package allows the user to specify and simulate counterfactual data under various interventions (e.g., static, dynamic, deterministic, or stochastic). These interventions may represent exposures to treatment regimens, the occurrence or non-occurrence of right-censoring events, or of specific monitoring events. The package also enables the computation of a selected set of user-specified features of the distribution of the counterfactual data that represent common causal target parameters (*the gold standards*), such as, treatment-specific means, average treatment effects and coefficients from working marginal structural models. In addition, the package provides a flexible graphical component that produces plots of directed acyclic graphs (DAGs) for observed (or post-intervention) data generating distributions.

We note that one of the distinguishing features of **simcausal** is that it allows the user to define and evaluate a causal target parameter, such as the ATE, that can then serve as the model-free gold standard. That is, the causal parameter is always the same functional of the counterfactual data distribution, regardless of the user-selected parameterization of the SEM. For example, the gold standard defined in this manner provides an objective measure of bias that does not depend on the modeling assumptions of a specific statistical method. Furthermore, coupled with a wide variety of possible data generating distributions that may be specified in **simcausal**, this package provides statisticians with a powerful tool for testing the validity and accuracy of various statistical methods. For example, one may use our package for validating an implementation of a novel statistical method, using the simulated data with the known truth (the true value of the causal parameter), prior to applying such an algorithm to real data, in which this truth is unknown. As another example, one may use **simcausal** to simulate data from a large variety of data-generating distributions and conduct a simulation study comparing the properties of different statistical procedures (e.g., bias, mean-squared error (MSE), asymptotic confidence interval coverage), using the user-selected causal parameter as the gold standard.

We also demonstrated the functionality of the package with a single time point intervention simulation study in Section 2.3 and a complex multiple time point simulation study in Section 2.4. Moreover, we also showed two real-world applications of the **simcausal** package in Sections 2.4 and 2.5, by replicating some of results of the two previously published simulation studies [78, 79, 68]. The first simulation study by [78] was initially conducted as a complement to a real data analysis in order to validate the claimed theoretical benefits of a new estimator in a simulated setting that was designed to resemble the data structure collected and used in the real-world study. The second simulation study by [68] evaluated the impact of the model misspecification of the treatment mechanism on *the MSE for the inverse probability-weighting (IPW) estimator, where the coefficients of the marginal structural model were used as the target causal quantity.* We note that in both of these instances, we were able to use **simcausal** to specify the desired data-generating distribution, then simulate repeated observed data samples, and finally, specify and evaluate the different causal parameters that were used in these simulation studies. We also note that the **simcausal** package vignette [115] contains additional replication results of the simulation study described by [78] that evaluated the comparative performance of targeted minimum loss based estimation (TMLE) and IPW estimation of a causal risk difference between two dynamic treatment regimens.

Finally, we note that the **simcausal** package is being actively developed and contains several new features that are beyond the scope of this chapter. In particular, recently implemented functionality allows one to simulate dependent observations using networks [34]. We refer to the *forthcoming* **simcausal** network vignette for details describing this new feature. We also note that the implementation of additional functionalities in future releases of the `simcausal` package should further expand its utility for methods research. Among such possible improvements is the evaluation of additional causal parameters, e.g., the average treatment effect on the treated [56, 58, 112], survivorship causal effects [60, 47] and direct/indirect effects [87, 92, 126, 131, 50].

2.7 Acknowledgments

FUNDING ACKNOWLEDGEMENT: This study was partially funded through internal operational funds provided by the Kaiser Permanente Center for Effectiveness & Safety Research (CESR). This work was also partially supported through a Patient-Centered Outcomes Research Institute (PCORI) Award (ME-1403-12506) and an NIH grant (R01 AI074345-07).

DISCLAIMER: All statements in this report, including its findings and conclusions, are solely those of the authors and do not necessarily represent the views of the Patient-Centered Outcomes Research Institute (PCORI), its Board of Governors or Methodology Committee.

Chapter 3

Simulation Studies in Network Data

3.1 Introduction

This chapter is concerned with design and implementation of simulation studies for network-dependent data. We describe a comprehensive set of tools implemented in the **simcausal**[1] R package [115], which allow simulating the types of dependent data one might collect on a community connected by a social or geographical network. The main purpose of this simulation tool is to study the validity of causal inference methods in such network-dependent data settings. We note that statistical estimation of causal effects in such connected settings is challenging, since the frequently made assumption of independence among units is generally invalid. Consider a setting with single time point exposure where we collect data on the baseline covariates, the exposures and the outcomes on N units. When these units are also connected by a network we might expect that the interactions between any two connected units can cause the exposure of one unit to have an effect on the outcome of the other unit - an occurrence often referred to as interference or spillover [57, 113]. Moreover, it is possible that both, the outcome and the exposure of one unit, are dependent on the baseline covariates of other connected units. We note that the new tools implemented in the **simcausal** packages are specifically designed for specification and simulation of the above described types of dependencies among units. In more detail, the **simcausal** process of specifying the distribution of N connected units generally consists of two steps. First, one describes the network of connections between these units (e.g., social or geographical network) by specifying either a network graph or a probabilistic network graph model for N nodes. Next, one specifies the distribution of the unit-level covariates (node attributes) by parameterizing a structural equation model (SEM) for connected units [63]. This SEM allows the covariates of one unit to be dependent on the covariates of other connected units via some known functional form which is controlled explicitly by the user. For this purpose, we developed a novel R interface which simplifies the specification of complex network-based functional

[1] **simcausal** package was developed using the R system for statistical computing [96] and it is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=simcausal`.

relationships between such units. Moreover, the network-based syntax can be *combined* with the existing syntax for specifying longitudinal data structures, allowing for simulations of network-based processes that also evolve in time. We refer to the vignette "*simcausal Package: Simulations with Complex Longitudinal Data*" [115] for general information about the **simcausal** package and the description of its syntax for simulating longitudinal IID data.

The past decade has seen an increasing body of literature devoted to estimation of causal effects in network-dependent data. Many of these studies seek to answer questions about the role of social networks on various aspects of public health. For example, Christakis et al. used the observational data on subjects connected by a social network to estimate the causal effects of contagion for obesity, smoking and a variety of other outcomes [27, 26], finding that many of these conditions are subject to *social contagion*, e.g., in one of the studies the authors found that the person's risk of becoming obese increases with an additional obese friend, even when one controls for all *measured* confounding factors. However, the statistical methods employed by these studies have come under scrutiny due to possibly anti-conservative standard error estimates that did not account for network-dependence among the observed units [70], and possibly biased effect estimates that could have resulted from: model misspecification [70, 128], network-induced homophily [111], and unmeasured confounding by environmental factors [111]. Given the potential high impact of such studies on future policy and public health decisions, it is important to be able to verify the robustness of statistical methods employed in these and similar studies. Thus, there is an emerging need for methods which can test the validity of various statistical methods for social network data and we argue that one of the possible approaches involves the conduct of simulation studies. That is, a practical way to test the validity of a certain statistical method involves its application against a large set of plausible data generating scenarios, specifically in relation to the types of dependent data one might see in social networks. Moreover, a carefully designed simulation study can test the method's sensitivity to violations of its key assumptions, highlight its limitations for specific types of data, and provide an important proof of concept that complements the results based on statistical theory. Finally, simulations can be also helpful in identifying errors in the implementation of complex statistical algorithms.

Network simulation studies have been previously applied to assess the validity of different estimation approaches in causal inference (for example, see [81], [34] and [5]). Such simulation studies have also been used as a guiding tool for comparison of the benefits of different experimental design strategies in network settings [133, 2, 53, 9]. However, to the best of our knowledge, there is no open-source simulation software for conducing network-based causal inference research that is similar to the tools and syntax implemented in the **simcausal** package. The main contribution of **simcausal** is in its novel interface which simplifies the design of simulation studies based on the non-parametric structural equation model for connected units. While the package provides the user with a broad range of possible data-generating distributions, it is specifically targeted towards causal inference research on the types of observational data that might be collected when observing members of a single social network. In other words, **simcausal** allows specification of a model based on a causal Directed Acyclic Graph (DAG), and in conjunction with previously specified network model,

this causal DAG can be used to define ways in which connected units depend on each other. Moreover, any R package that can simulate such network graphs can be used with **simcausal**. For example, in this chapter we use the **igraph** R package [29] for network graph simulations.

In summary, these are some of the advantages and differences of our proposed network simulation package over other open-source tools. **simcausal** provides a simple and concise interface for specifying a network data-generating distribution and incorporating the network structure into various forms of functional dependence of one unit on other *connected* units. It permits the simulation of such interconnected data structures, as well as the generation of the counterfactual data under single or multiple time-point interventions. The user-defined causal parameters can then be evaluated from the counterfactual data and serve as gold-standards in testing the validity of various statistical methods. Finally, the **simcausal** package provides the syntax for specifying complex longitudinal data structures *in combination* with the network-dependent syntax, allowing one to simulate complex network processes that also evolve over time.

Other related R packages

As of May 29, 2016, there were 208 R packages on CRAN that contain the word "network" in their title sentence. A large number of these packages are dedicated towards visual analysis and representation of networks (e.g., **ggnetwork** [19], **igraph** [29], **d3Network** [40]), re-constructing the network based on biological, neural and other field-specific data (e.g., **interventionalDBN** [117], **LogitNet** [134], **RSNNS** [11], **dna** [42]), statistical analysis of networks based on specific network-generating models (**multiplex** [100], **lvm4net** [44]). In addition, a large collection of packages often referred to as a "**statnet** suite of packages" provides various tools for social network analysis, visualization, simulation and diagnoses based on the statistical methods of exponential-family random graph models (ERGMs) or other related parametric model families for networks (e.g., **statnet** [52], **sna** [23], **EpiModel** [59], **ergm** [51], **tergm** [62]). Other packages for statistical analysis of network or network-related data include, among others, **netdiffuseR** [132], **nets** [21], **ebdbNet** [97] and **tmlenet** [116]. Finally, a large number of R packages are targeted towards analyses and simulation of various networks, network evolution over time and the modeling various network features, such as the creation of the tie between two nodes and answering questions such as, how and why certain network ties are formed? Among the packages that are tailored to such analyses are **CIDnetworks** [1], **tsna** [10], **networkDynamic** [24] and **egonet** [110]. Another class of packages that is worth noting are those specifically targeted towards modeling of epidemics on a network graph, such as packages **epinet** [48], **netdiffuseR** [132], **EpiModel** [59], for example, by relying on the agent-based modeling or ERMG techniques. In addition, the following packages are specifically designed for simulations of the network-graph data: **SocialMediaLab** [46], which provides tools for collecting and generating social media data for networks; **hybridModels** [109], which allows simulations of stochastic models for transmission of infectious diseases in longitudinal networks; **NetSim** [118], a package for simulating

social networks; and finally, **RSiena** [99], a package designed for simulation-based analysis of networks as well as model fitting for longitudinal network data.

Organization of this chapter

The rest of this chapter is organized as follows. In Section 3.2, we formally describe our assumed observed data structure and provide the technical definition of the underlying nonparametric structural equation model (NPSEM) for connected units. In Section 3.3 we provide some technical details of the **simcausal** interface for simulating data according to the user-specified parameterization of this NPSEMs. In Section 3.4, we describe an example of a simulation that uses a network distributed according to the Erdos-Renyi model [98]. In Section 3.5, we describe the use of the package for simulating from more small world network distribution [135]. We also illustrate the use of **simcausal** in a typical simulation study, evaluating the true sample-average mean causal outcome (*the gold-standard*) of a single time-point stochastic intervention on continuous exposure. In Section 3.6, we demonstrate how one of our network simulation examples could be used in practice for assessing the performance of statistical methods for estimation of causal effects among network-dependent observations. In Section 3.7, we simulate a network in a longitudinal setting in which effects are propagated over time and across units. We conclude with a discussion in Section 3.8.

3.2 Technical details

We start by introducing some notation. Suppose that we can simulate a sample of N connected observations, where each observed unit is indexed as $i = 1, \ldots, N$. We let $F_i \subseteq \{1, \ldots, N\} \backslash i$ denote the set of observations that are assumed "*connected*" to i or, as we will refer to it, the units in F_i are "*friends*" of i. In other words, we assume that each set F_i consists of a unique set of indices j in $\{1, \ldots, N\}$, except for i itself. We also allow F_i to be empty, which would imply that observation i is not receiving input from any other units. We assume that i may receive input from other observations only if those observations are listed as part of F_i. We will refer to the union of all F_i as a "*network profile*" on all N observations, which will be denoted by $\mathbf{F}$. Let $O_i = (W_i, A_i, Y_i)$ denote the data collected on each observation i, for $i = 1, \ldots, N$, where W_i denotes all the baseline covariates for i, A_i denotes the exposure of i and Y_i denotes the outcome of i. Let $\mathbf{W} = (W_i)_{i=1}^N$, $\mathbf{A} = (A_i)_{i=1}^N$, $\mathbf{Y} = (Y_i)_{i=1}^N$ and $\mathbf{O} = (\mathbf{W}, \mathbf{A}, \mathbf{Y})$. Finally, we assume $F_i \in W_i$, that is, the network of friends of i is assumed to be part of i's baseline covariates.

Nonparametric structural equation model for connected units

We now define the nonparametric structural equation model (NPSEM) [86] for N connected units, similar to that described in [63]. This model will form the basis for describing the type of network-based data generating distributions that can be defined within the **simcausal**

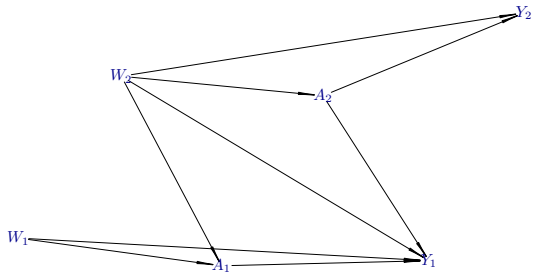

Figure 3.1: An example of a directed acyclic graph (DAG) for two observations, where unit 1 is dependent on unit 2, but not vice-versa.

package. To help with the presentation, we first consider a *more general* NPSEM, which is then followed with the definition of our actual NPSEM of interest. Suppose that the N connected observations are generated by applying the following, general, NPSEM[2]:

$$\begin{aligned} W_i &= f_{W_i}(U_{W_i}), & i=1,\ldots,N, \\ A_i &= f_{A_i}(W_i,(W_j : j \in F_i),U_{A_i}), & i=1,\ldots,N, \\ Y_i &= f_{Y_i}(A_i,W_i,(A_j,W_j : j \in F_i),U_{Y_i}), & i=1,\ldots,N, \end{aligned}$$

where for now we assume that the error terms $(U_{W_i},U_{A_i},U_{Y_i})$ are sampled as IID, for $i=1,\ldots,N$. Note that the above NPSEM could be represented as a Directed Acyclic Graph (DAG) [85], by drawing arrows from causes to their effects. However, such a DAG would have to include all N dependent observations (the entire network), since the above NPSEM includes a separate equation for each observed unit $i=1,\ldots,N$. For example, for a network with two units ($N=2$), in which unit $i=1$ is dependent on unit $i=2$, but not vice-versa, the NPSEM could be depicted with a DAG shown in Figure 3.1. We also note that the error terms $(U_{W_i},U_{A_i},U_{Y_i})$, for $i=1,2$, are excluded from this causal DAG [88], with the implication that each of the remaining variables is subject to the influence of its own independent error.

To simplify our notation and to also to maintain the grip on the increasing dimensionality of such network-connected data, we assume that there are some known summary measures:

$$W_i^s := w_i^s(W_i,W_j : j \in F_i)$$

and

$$A_i^s := a_i^s((A_i,W_i),(A_j,W_j : j \in F_i)),$$

and we use the short-hand notation W_i^s for $w_i^s(\cdot)$ and A_i^s for $a_i^s(\cdot)$. We assume that these summary measures W_i^s and A_i^s are of constant-in-i dimension that does not depend on N.

[2]This NPSEM is general in the sense that it makes no functional restrictions on f_{W_i}, f_{A_i} and f_{Y_i}, allowing each i-specific set of covariates (W_i,A_i,Y_i) to be generated from their own i-specific functions. We do, however, assume that the dependence of i on other units is limited to set F_i and we also make an assumption of independent errors $(U_{W_i},U_{A_i},U_{Y_i})$ across units.

We now assume that the observed N units are actually generated from the following NPSEM, which can be viewed as a special case of the NPSEM presented earlier:

$$\begin{aligned} W_i &= f_{W_i}(U_{W_i}), & i=1,\ldots,N, \\ A_i &= f_A(W_i^s, U_{A_i}), & i=1,\ldots,N, \\ Y_i &= f_Y(A_i^s, W_i^s, U_{Y_i}), & i=1,\ldots,N. \end{aligned}$$

This NPSEM implies that the observed data could be simulated in the following manner:

1. Start by generating N baseline covariates $(W_1,\ldots,W_N)$, by first drawing the (independent or weakly dependent) errors U_{W_i} and then applying the equations $f_{W_i}(U_{W_i})$, for $i=1,\ldots,N$;
2. Generate N baseline summaries $(W_1^s,\ldots,W_N^s)$, by applying the baseline summary measures $w_i^s(\cdot)$ to W_i and $(W_j : j \in F_i)$, for $i=1,\ldots,N$;
3. Generate N exposures $(A_1,\ldots,A_N)$, by first drawing the errors U_{A_i} and then applying the common equation $f_A(\cdot)$ to each W_i^s and U_{A_i}, for $i=1,\ldots,N$;
4. Generate N exposure summaries $(A_1^s,\ldots,A_N^s)$, by applying the exposure summary measures $a_i^s(\cdot)$ to (A_i,W_i) and $(A_j,W_j : j \in F_i)$, for $i=1,\ldots,N$; and
5. Generate N outcomes $Y_1,\ldots,Y_N$, by drawing the errors U_{Y_i} and then applying the common equation $f_Y(\cdot)$ to (A_i^s,W_i^s) and U_{Y_i}, for $i=1,\ldots,N$.

Note that in this chapter we only consider NPSEMs which assume a certain independence structure of the errors, namely, conditional on $\mathbf{W}$ we assume that: (1) (U_{A_i},U_{Y_i}), for $i=1,\ldots,N$ are IID; and (2) For each i, U_{A_i} is independent of U_{Y_i}. The above structural equation model assumptions on generating A_i and Y_i imply that if two different units with the same number of friends have the same individual covariate and treatment values, and also have the same values for the covariates and treatments of their friends, they will be subjected to the same conditional distribution for drawing their treatment and outcome. We note that we had assumed U_{W_i} are iid and since f_{W_i} are allowed to be different for each i this corresponds with assuming that $W_1,\ldots,W_N$ are independent, but not necessarily identically distributed. However, our package also allows simulating dependent $W_1,\ldots,W_N$, for example, by defining a *latent* (hidden) covariate W_i^* which is shared between all observations connected to i and then defining the *observed* baseline covariate for i and all $j \in F_i$ conditionally on W_i^*. Such simulation procedure allows one to introduce dependence between the observed W_i and W_j, whenever i and j are connected.

These independence assumptions on $U_i := (U_{W_i},U_{A_i},U_{Y_i})$, for $i=1,\ldots,N$ imply that:

1. $W_1,\ldots,W_N$ are independent (or more generally, their dependence is weak enough);
2. Conditional on $\mathbf{W}$, the random variables $(A_1,\ldots,A_N)$ are mutually independent; and
3. Conditional on $(\mathbf{A},\mathbf{W})$, the random variables $(Y_1,\ldots,Y_N)$ are mutually independent.

Thus, all the dependence between units is explained by the observed pasts of the units themselves and of their friends. Finally, these structural assumptions lead to the following

assumptions for the conditional distribution of Y_i and A_i. Specifically, let $P(A_i \,|\, \mathbf{W})$ denote the conditional distribution of the exposure A_i, given all $\mathbf{W}$; and let $P(Y_i \,|\, \mathbf{A}, \mathbf{W})$ denote the conditional distribution of the outcome A_i, given all $(\mathbf{A}, \mathbf{W})$, for observations $i = 1, \ldots, N$. The above NPSEM with the network structure $\mathbf{F}$ imply the following for these conditional distributions:

1. Each $P(A_i \,|\, \cdot)$ depends on $(W_i, W_j : j \in F_i)$ only as a function of some fixed-dimension summary measure $w_i^s(W_i, W_j : j \in F_i)$; and
2. Each $P(Y_i \,|\, \cdot)$ depends on (A_i, W_i) and $(A_j, W_j : j \in F_i)$ only as a function of some fixed-dimension summary measures $a_i^s((A_i, W_i), (A_j, W_j : j \in F_i))$ and $w_i^s(W_i, W_j : j \in F_i)$.

Note that the above-defined NPSEM also implicitly encodes the definition of counterfactual variables, i.e., variables which would result from some particular interventions on a set of endogenous variables. For example, for a particular vector of treatment assignments $\mathbf{a} = (a_1, \ldots, a_N)$, we could modify the NPSEM as follows:

$$\begin{array}{lcll} W_i & = & f_{W_i}(U_{W_i}), & i = 1, \ldots, N, \\ A_i & = & a_i, & i = 1, \ldots, N, \\ Y_{i,\mathbf{a}} & = & f_Y(a_i^s(\mathbf{a}, \mathbf{W}), W_i^s, U_{Y_i}), & i = 1, \ldots, N, \end{array}$$

where the equations for W_i were kept unchanged, each A_i was set to a_i, and each $Y_{i,\mathbf{a}}$ denotes the counterfactual outcome of unit i, under the network intervention on all N units that sets $\mathbf{A} = \mathbf{a}$. In this chapter, we will refer to $(\mathbf{W}, \mathbf{a}, \mathbf{Y_a})$ as *counterfactual data* for N units and we define our target causal parameter as a function of such counterfactual data distribution. For example, the average treatment effect (ATE) can be simply expressed as $E[\mathbf{Y_1} - \mathbf{Y_0}]$.

3.3 Using simcausal for simulating networks and network-dependent data

To define the distribution of the data (and to simulate such data), **simcausal** uses a `DAG` object, which will typically consist of a collection of individual *nodes* (random variables). Such nodes are defined each time the user calls the `node()` function. That is, each call to `node()` defines the conditional distribution(s) of either a single or time-varying node. Collectively, these `node()` calls define a single `DAG` object, which parametrizes the distribution of the data that the user wants to simulate. By default, the distribution of any new node for a single observation i can be dependent only on i's values of the previously defined nodes, and not the node values of other observations. As a result, any two observations sampled from such `DAG` object will be independent. However, there are many real-life examples when one wishes to simulate data based on some either a priori known or hypothesized information about the network of connections between different observations. The new functionality we

describe here allows the user to specify the distribution of a network graph and then use that network to define the distribution of covariates for each observation i conditionally on the covariates of other observations. For example, if we assume that the observation j is part of i's social or geographical network (or as we will call it, the observation j is a *friend* of i), then the nodes of j are allowed to influence the distribution of the nodes of i. Moreover, we assume that the functional form of such dependence can be described by some user-specified network-based summary measures. For that purpose, we used the R list subsetting operator "`[[...]]`" and re-defined it specifically for indexing and building of such network summaries based on observations that are friends of i (e.g., "`Var[[indx]]`"). We also note that **simcausal** does not allow simultaneous friend references of the same node, that is, each newly added node can be defined only as a function of the nodes that have been previously added to the `DAG` object.

Defining the network

In order to perform network-based simulations with the **simcausal** package, the user has to declare a function which will return a specific network matrix, and we will refer to any such function as the *network generator*. In particular, a network generator is any user-specified function that returns a network profile $\mathbf{F}$ on N observations, defined as a matrix of N rows, each row i containing the set of friends F_i of observation i, i.e., row i consists of a vector of observations from $\{1, \ldots, N\}$ that are assumed connected to i. The network generator function should accept at least one named argument, `n`, and it must return a network matrix with `n` rows and `Kmax` columns, where `Kmax` stands for a maximal possible number of friends for any observation. When observation i happens to have fewer than `Kmax` friends, i.e., $|F_i| <$`Kmax`, the remainder of the matrix row i must by filled with `NA` (missing) values. Note that an observation i is allowed to have no friends, which is denoted by an empty friend set F_i, in which case the row i of the network matrix should only consist of `NA` (missing) values.

Once such a network generator has been defined, the next step is to *add* this network to a specific `DAG` object. This is accomplished by simply calling the `network` function, specifying the name of the network generator as its argument "`netfun`" and adding this network function call to the current `DAG` object with the R operator "`+`". In other words, the `network` function call defines the network and is added to an existing `DAG` object with a syntax "`+network(...)`". Note that this is identical to the **simcausal** syntax for adding new `node` function calls to a growing `DAG` object when defining data for IID observations. This network can then serve as a backbone for defining the dependent-data structural equation models within such a `DAG` object. More specifically, the previously defined node values of i's friends can be now referenced as part of the new `node` function calls, defining the conditional distribution of the node for each observation $i = 1, \ldots, N$. The examples following this section illustrate this functionality for various network models.

Finally, we note that the `network` function can accept any number of user-specified optional arguments, where each of such optional arguments must be an evaluable R expression. These expressions will not be evaluated until the data simulation step, at which point all

of them are passed as arguments to the function that defines the network generator. Thus, just like the regular `node` function expressions, the `network` function expressions can refer to any standard or user-specified R functions and can reference any of the previously defined `DAG` nodes (i.e., `DAG` nodes that were added prior to `network` function call). This feature can be useful when, for example, one wishes to simulate the network in which the probability of forming a tie between two units depends on the previously simulated unit-specific variable values (such as the baseline risk factors on each unit).

Using the syntax `[[...]]` for network-based variable subsetting

Following the `network` function call, subsequent calls to `node` function can employ our repurposed list subsetting operator "`[[...]]`" for indexing the node values of friends. First, the variable which is to be used for network subsetting is specified in front of the subsetting operator, e.g., "`A[[...]]`". Second, the friend values of the variable `A` are specified by the subsetting index, e.g., "`A[[1:5]]`". This expression will look up the values of node `A` for friends indexed from 1 to 5 and it will be evaluated for all observations $i = 1, \ldots, N$. The specific ordering of friends is determined by the column ordering of the network matrix returned by the network generator. Such network-indexing expressions can be also used as inputs of different R functions, enabling evaluation of various network-based summaries. For example, the expression "`sum(A[[1:Kmax]])`" will specify a vector of length N that will consist of a sum of `A` values among all friends of i, for each observation $i = 1, \ldots, N$. This syntax is fully generalizable towards any function which can operate on matrices, such as the matrix result of the expression "`A[[1:Kmax]]`". Moreover, two of the commonly used R functions, `sum` and `mean`, are automatically replaced with their row-based counterparts: the functions `rowSums` and `rowMeans`. Thus, the expressions "`sum(A[[1:Kmax]])`" and "`rowSums(A[[1:Kmax]])`" can be used interchangeably.

We illustrate this syntax with a simple example. Suppose that we have a `DAG` object, named "`D`" and we use the network generator, named "`rnet.gnp`"[3]. As shown in the code snippet below, we first define an empty `DAG` object `D`, then we add the network named "`net`" to object `D` and we also define an IID Bernoulli variable named "`Var`".

```
library("simcausal"); library("magrittr")
D <- DAG.empty() +
  network("net", netfun = "rnet.gnp", p = 0.1) +
  node("Var", distr = "rbern", prob = 0.5)
```

Next, we define a new node, named "`Var.F1`", as the value of `Var` for the first friend of each observation $i = 1, \ldots, N$. We do this by defining a node with a constant (degenerate) distribution, `distr="rconst"`, and indexing the first friend of each observation with the expression "`Var[[1]]`", as shown in the following example:

[3]Note that the network generator `rnet.gnp` is provided as part of the **simcausal** package. It uses the **igraph** R package function `sample_gnp` to sample a network graph and then converts its output into the **simcausal** network matrix representation. See `?rnet.gnp` for additional information.

```
D <- D + node("Var.F1", distr = "rconst", const = Var[[1]])
```

Suppose we now wish to list the `Var` values among the first 4 friends from each set F_i. This can be accomplished by defining a single multivariate node and using the expression "`Var[[1:4]]`", as shown next:

```
D <- D + node(paste0("Var.F",1:4), distr = "rconst", const = Var[[1:4]])
```

As our final example, we define a new degenerate node, named "`mean.F1to4`", as the mean of `Var` amongst the first 4 friends, for each $i = 1, \ldots, N$. We also define a new Bernoulli node, named "`Var.F1to4`", for which the i-specific probability of success is also given as the mean of `Var` amongst the first 4 friends in F_i. The data for the variables defined thus far can now be simulated by simply calling the functions `set.DAG` and `sim` in sequence and specifying the desired sample size N with the argument "`n`".

```
{ D <- D +
  node("mean.F1to4", distr = "rconst", const = mean(Var[[1:4]], na.rm=TRUE)) +
  node("Var.F1to4", distr = "rbern", prob = mean(Var[[1:4]], na.rm=TRUE)) } %>%
set.DAG(n.test = 50) %>%
sim(n = 50)
```

In summary, for a given indexing vector `indx`, the network-indexing expression, such as "`Var[[indx]]`", will evaluate to a matrix with N rows and the number of columns determined by the length of `indx`. We note that indexing variable `indx` can be a non-negative integer-valued vector, with values starting from 0 and bounded above by a special reserved constant named "`Kmax`". That is, the variable `Kmax` can be used for finding out the maximal friend index for any given network, as we demonstrate in examples in following sections. In addition, one can use `0` as part of the same friend indexing vector, where the expression "`Var[[0]]`" is equivalent to using "`Var`". This provides a convenient syntax for indexing the actual `Var` value of observation i along with the `Var` values of i's friends, for example, allowing the expressions such as "`sum(Var[[0:Kmax]])`". Furthermore, for a specific observation i, the expression "`Var[[k]]`" will evaluate to a missing value (i.e., "`NA`") whenever i has fewer than `k` friends. This default behavior can be also altered by passing a special named argument "`replaceNAw0=TRUE`" to the corresponding `node()` call, in which case all of such missing (`NA`) values are automatically replaced with `0` values. In addition, any node expression can reference a special reserved variable "`nF`", which is a vector of length `n` and it stores the total number of friends for observation j in its jth entry.

Finally, we note that any function that defines a network-indexing node summary can be similarly applied as a summary of a time-varying node and vice-versa. For example, for a time varying node "`Var.t`", the expression "`sum(Var.t[t_indx])`" is analogous to our previous example of the network summary "`sum(Var[[indx]])`", except that in the former case we are summing the values of a time-varying node `Var.t` for time-points defined by the

indexing vector `t_indx`. We also provide some additional examples of such network summaries in the following sections. However, for more in-depth description of this functionality we refer to Section "*Defining node distributions and vectorizing node formula functions*" of the **simcausal** package vignette "*simcausal Package: Simulations with Complex Longitudinal Data*". Furthermore, both of these indexing operators, i.e., the time indexing operator "`[...]`" and the network indexing operator "`[[...]]`", can be combined to form a single summary applied to a time-varying node, as we demonstrate in Section 3.7.

3.4 Simulation with Erdos-Renyi network

Specifying the structural equation model for dependent data

We describe a simulation study of a hypothetical community of intravenous drug users, considered to be under a high risk for HIV infection. We suppose that the exposure of interest was defined as the indicator of receiving antiretroviral therapy while the outcome was defined as the HIV status, which was assessed after some pre-specified follow-up period. We also node that this simulation set-up is later extended to a more realistic setting with longitudinal data in Section 3.7.

For this simulation the network graph is sampled according to the network generator called "`rnet.gnm`", which is provided as part of the **simcausal** package. This network generator uses the `sample_gnm` function of the `igraph` package [29] to sample a directed random graph according to Erdos-Renyi model [98]. The nodes of the graph returned by the `sample_gnm` function are treated as individual observations and are indexed as $i = 1, \ldots, N$. A directed edge from node j pointing to node i implies that the unit j is a friend on unit i, and hence $j \in F_i$, but not vice-versa, i.e., $i \in F_j$ only if there is a separate directed edge from node i to node j. The function `rnet.gnm` converts this network graph to a matrix of network IDs, where each friend set F_i (matrix row i) is determined according to the above described rule. Finally, function `rnet.gnm` accepts two arguments: `n` - the total sample size (number of nodes in a network graph), and `m_pn` - the total number of edges in the network as a fraction of `n`. Note that the argument `n` needs to be the first argument of any network generator, while the argument `m_pn` is optional.

We proceed by first instantiating an empty `DAG` object, which is assigned to variable "`D`" below. We will use this `DAG` object to define the network distribution, as well as to encode the unit-specific distribution of the data. As a next step, we then register the network generator function (`rnet.gnm`), making it a part of the object `D`. This will allow us to: (a) generate a specific network, and (b) use this network as a way to connect nodes defined within the same `DAG` object `D`. Specifically, as we show in the following code snippet, we add a network to the object `D` by using syntax "`D<-D+network()`". The name of the network generator is passed on to the `network()` call with an argument "`netfun`". Note that the `network()` call also requires a "`name`" argument (first argument), which specifies a name of a particular network[4].

[4]The actual value of the `name` argument of `network()` function becomes only relevant when the user

Note that the function `network` allows passing any number of additional arguments, however all of these arguments must be named. These arguments will be passed on to the network generator function (e.g., `rnet.gnm`) during the process of data simulation. Note that one can use such arguments for additional parameterization of the network distribution, such as the argument "`m_pn`" in the example below.

```
library("simcausal")
D <- DAG.empty() + network(name = "ER.net", netfun = "rnet.gnm", m_pn = 5)
```

As a next step, we define an integer node named "`nF`", a vector with its jth entry set equal to the total number of friends for unit j. Note that the special reserved variable `nF` is automatically evaluated by the package and thus does not need to be explicitly defined as a separate node.

```
D <- D + node("nF", distr = "rconst", const = nF)
```

In our next example we define three IID covariates $W = (W(1), W(2), W(3))$, where $W(1)$ is denoted as a node "`W1`" and it defined a categorical baseline risk factor, $W(2)$ is denoted as a node "`W2`" and it defines a binary confounder positively correlated with `W1`, and $W(3)$ is denoted as a node "`W3`" and it defines a binary indicator of having an HIV infection at baseline.

```
D <- D +
  node("W1", distr = "rcat.b1",
        probs = c(0.0494, 0.1823, 0.2806, 0.2680, 0.1651, 0.0546)) +
  node("W2", distr = "rbern", prob = plogis(-0.2 + W1/3)) +
  node("W3", distr = "rbern", prob = 0.05)
```

We are finally ready to define the network summaries which will use the network. Specifically, in the example below we introduce three network-based summaries `sumW1`, `sumW2` and `sumW3`, defined as the sums of the unit-specific baseline covariate values of the unit's friends. For example, when the user simulates `n` observations, the variable `sumW1` will evaluate to a vector of length `n`, with the ith entry in `sumW1` being defined as $\sum_{j \in F_i} W_j(1)$, which corresponds with the sum of the values of `W1` among all friends of unit i[5]. Note that `sumW3` thus represents the number of friends of each observation who were infected at baseline. The summing order is based on the ordering of the columns in the network matrix returned by the network generator. The last friend index is indicated with `Kmax`, where `Kmax` is a

wants to either over-write the existing network or use several different networks within the same `DAG` object.

[5]As previously described, the expression "`sum(W1[[1:Kmax]])`" is automatically converted to an expression `rowSums(W1[[1:Kmax]])`, thus, correctly evaluating the unit-specific sums of the corresponding input matrix rows, such as, `W1[[1:Kmax]]`. However, this functionality does not apply to any other functions besides `sum` and `mean`. It is the responsibility of the user to provide a function which will appropriately handle the matrix evaluation result of `W1[[1:Kmax]]`.

constant reserved by **simcausal** and it is equal to the maximal number of friends for any unit. As previously described, when the unit j happens to have fewer than `Kmax` friends, the default rule is to return `NA` for such non-existing friend covariate values. However, these none-existing covariate values will be automatically replaced with `0` whenever an additional argument `replaceNAw0 = TRUE` is passed to `node()` call, as we show in the examples below:

```
D <- D +
  node("sumW1", distr = "rconst", const = sum(W1[[1:Kmax]]), replaceNAw0 = TRUE) +
  node("sumW2", distr = "rconst", const = sum(W2[[1:Kmax]]), replaceNAw0 = TRUE) +
  node("sumW3", distr = "rconst", const = sum(W3[[1:Kmax]]), replaceNAw0 = TRUE)
```

Suppose now that we also wanted to obtain the covariate values for specific friends, e.g., the values of covariate `W1`. Below we provide such an example, by defining one multivariate node (5 columns) which contains the values of `W1` among the first 5 friends. Note that we are no longer using the argument `replaceNAw0 = TRUE`, which should set the evaluation result to `NA` whenever the corresponding friend of rank `k` does not exist.

```
D <- D + node(paste0("W1.F",c(1:5)), distr = "rconst", const = W1[[1:5]])
```

Next, we define the conditional distribution for the binary exposure A, denoted with node "`A`" below (e.g., indicator of receiving antiretroviral therapy). The probability of success for A is defined as a logit-linear function of `W2` and the above network-based summaries `sumW1`, `sumW2` and `sumW3`.

```
D <- D +
  node("A", distr = "rbern",
    prob = plogis(2 - 4*W2 - 0.1*sumW1 - 0.4*sumW2 + 1.5*sumW3))
```

Note that the above expression in `prob` argument of node `A` did not need to use the names of the network summaries `sumW1`, `sumW2` and `sumW3`. Thus, the node `A` could have directly defined the network summaries eliminating the need to define separate nodes `sumW1`, `sumW2` and `sumW3`, as we show the following alternative definition of `A`[6]:

```
D <- D +
  node("A", distr = "rbern",
    prob = plogis(2 - 4*W2 -
      0.1*sum(W1[[1:Kmax]]) - 0.4*sum(W2[[1:Kmax]]) + 1.5*sum(W3[[1:Kmax]])),
    replaceNAw0 = TRUE)
```

[6]Note that by adding another `node()` call with the same node name "`A`" we over-write the previously defined node `A` (where in this particular example the new node `A` happens to be equivalent to the one we defined previously)

In the following example we define the network-based summary measure "sumW3A", which involves an interaction of friend's exposures A and friend's baseline covariates W3. More specifically, we define the summary measure sumW3A for each unit i as $\sum_{j \in F_i} W_j(3)(1 - A_j)$, i.e., its the total number of i's friends who were infected at baseline (W3 was 1) and were also unexposed (A was 0).

```
D <- D +
  node("sumW3A", distr = "rconst",
    const = sum(W3[[1:Kmax]] * (1 - A[[1:Kmax]])),
    replaceNAw0 = TRUE)
```

Next, we define the conditional distribution of the binary outcome Y, denoted as a node "Y" below. We assume that Y is an indicator of adherence to the antiretroviral therapy throughout some pre-specified follow-up period. In the example below, we model this outcome as a function of the above described network summaries. Specifically, for each observation i, we assume that the probability of success for Y is a logit-linear function of the summaries sumW3, sumW3A and the baseline covariate value W2. We also add another network summary, making the outcome for observation i also dependent on the average exposure of i and the exposures of i's friends, i.e., $(A_i + \sum_{j \in F_i} A_j)/(|F_i| + 1)$. This new summary is defined by the expression sum(A[[0:Kmax]])/(nF+1)) in the example below. Also note that we are including 0 in the network subsetting of the node A (i.e., A[[0]]), which is equivalent to just using A.

```
D1 <- D +
  node("Y", distr = "rbern",
    prob = plogis(-1 + 0.5*A + 2*sumW3A -
                  2*sumW3 + 3*W2 + 0.25*sum(A[[0:Kmax]])/(nF+1)),
    replaceNAw0 = TRUE)
```

Note that, just like with an alternative definition of the node A above, we could have also defined all of the above network summaries directly inside the definitions of node Y. Finally, we finish specifying the observed data-generating distribution by calling the function set.DAG(D), as shown below:

```
Dset1 <- set.DAG(D1, n.test = 100)
```

Simulating network and observed data

Having defined the network-based distribution of the data with a specific DAG object, one can simulate data by simply calling the function sim, as we show in the example below. Here we simulate 200 observations from the distribution defined by the above object Dset1.

```
datnet <- sim(Dset1, n = 100, rndseed = 543)
head(datnet, 2)
```

```
##   ID nF W1 W2 W3 sumW1 sumW2 sumW3 W1.F1 W1.F2 W1.F3 W1.F4 W1.F5 A sumW3A Y
## 1  1  6  3  1  0    18     4     0     1     3     1     4     5 0      0 0
## 2  2  3  2  0  0    10     2     0     3     3     4    NA    NA 1      0 0
```

We also note the presence of the missing values (`NA`) for some network covariates in the above simulated data frame (e.g., `W1.F4` and `W1.F5`). As previously stated, these missing network covariates imply that the unit has fewer friends than the index of the syntax `[[...]]` (e.g., `W1.F5=W1[[5]]`). The data frame returned by the `sim()` function can be also used for extracting the simulated network, as we show next. The data frame `datnet` contains an attribute called `netind_cl`, which is an `R6` object of class `NetIndClass`. This object is used for storing the network, as it was returned by the network generator. In particular, the field `NetInd` contains the network matrix and the field `nF` contains the vector with total counts of the simulated friends for each observation, i.e., $\mathbf{F} = (F_1, \ldots, F_N)$ (see `?NetIndClass` for more information).

```
NetInd_mat <- attributes(datnet)$netind_cl$NetInd
head(NetInd_mat, 2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
## [1,]    7   57   65   67   77   83   NA   NA   NA    NA    NA
## [2,]    1   60  100   NA   NA   NA   NA   NA   NA    NA    NA
```

```
nF <- attributes(datnet)$netind_cl$nF
head(nF, 2)
```

```
## [1] 6 3
```

Plotting network and SEM

The `plot.igraph` function in **igraph** package can be used for visualizing such simulated network. However, the network ID matrix `NetInd_mat` needs to be first converted back into its original **igraph** object representation (`g`), as we show below.

```
library("igraph")
g <- sparseAdjMat.to.igraph(NetInd.to.sparseAdjMat(NetInd_mat, nF = nF))
```

We can now use the `plot.igraph` function of the **igraph** package to visualize the simulated network structure stored in the object `g`, as shown in Figure 3.2.

Using more than one network in a single dataset or over-writing an existing network

Note that additional `network()` calls can be added to the same object `D`. When the same `name` argument in the new `network()` call is used, the old network definition is automatically

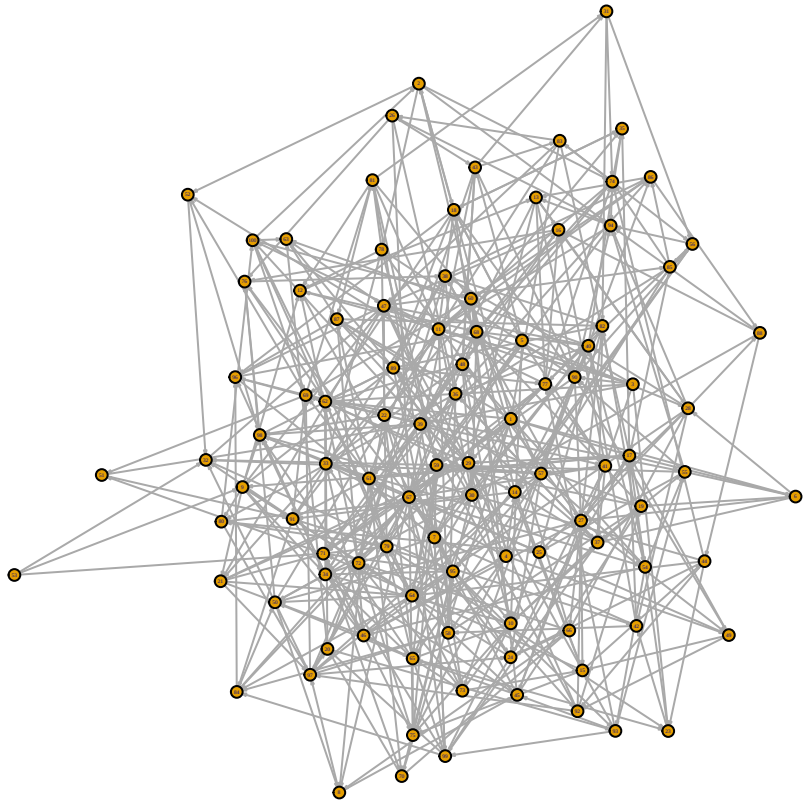

Figure 3.2: Example of a network sampled from the Erdos-Renyi model.

replaced by the new one. In the example below, we overwrite the previously defined network `"ER.net"` with another network, with the same network generator `rnet.gnm`, but with a different parametrization of the argument `m_pn`.

```
D <- D + network(name = "ER.net", netfun = "rnet.gnm", m_pn = 10)
Dset.alt <- set.DAG(D, n.test = 200)
datnet <- sim(Dset.alt, n = 200, rndseed = 543)
```

In contrast, when another `network()` call provides a different `name` argument, e.g., `name = "new.ER.net"`, both networks (old and new) will be present within the same `DAG` object and both networks can be used for constructing the network summaries. Specifically, any node with a network-based summary that is added to `D` *after* the network `new.ER.net`, will be computed on the basis of this new network, while all the network summaries defined previously within `D` *before* `new.ER.net` was added, will continue to use the old network. We illustrate this functionality with the following example, where we use the the same `DAG` object `D` and we add a new network named `new.ER.net`, which is then followed by a new network summary measure defined by the node "`new.sumW1`". Note that the summary defined in `new.sumW1` is exactly the same function as the previously defined summary `sumW1`, i.e.,

it is the sum of the friend values of the node `W1`. Thus, their unit-specific values should be equivalent when the two summaries use the same network structure. However, in our case the summary in `sumW1` is evaluated on the basis of the network `ER.net`, while the summary `new.sumW1` is evaluated on the basis of the re-sampled network `new.ER.net`. As the example below demonstrates, by using two different networks these two functionally equivalent summaries evaluate to different observation-specific values.

```
D <- D +
  network(name = "new.ER.net", netfun = "rnet.gnm", m_pn = 10) +
  node("new.sumW1", distr = "rconst", const = sum(W1[[1:Kmax]]),
    replaceNAw0 = TRUE)
Dset.alt <- set.DAG(D, n.test = 200)
datnet <- sim(Dset.alt, n = 200, rndseed = 543)
head(datnet[,c("sumW1", "new.sumW1")], 2)
```

```
##   sumW1 new.sumW1
## 1    50        17
## 2    36        32
```

We note that this functionality is also demonstrated in a longitudinal simulation study in Section 3.7.

3.5 Simulation with small world network, continuous exposure and a single time point stochastic intervention

The simulation study described here is a simplified version of a hypothetical observational study that might have been conducted to evaluate the social influence of healthy living on personal long term health status. We imagined that this study collected data on an interconnected community of N individuals. For each individual i, we would have measured their social network F_i, baseline covariates W_i, an exposure A_i and a binary outcome Y_i. The exposure was assessed as a continuous physical activity index and the outcome indicated if the person was obese after some follow-up period. We evaluated the average-causal effect of a stochastic intervention that intervened only on some members of the community by shifting their observed physical activity level by some known constant $shift > 0$, while keeping the exposures of others unchanged. More precisely, such stochastic intervention for each a unit i can be defined by a function $\delta(A_i, W_i)$ that assigns the new (intervened) exposure to either $A_i + shift$ or A_i, depending on the covariate values W_i. Such interventions have been described in the past [75] and they arise naturally in settings with continuous exposures where it is not feasible to intervene on every member of the population. For example, in our hypothetical study it might be infeasible to increase the level of physical activity for an

individual with a pre-existing medical condition who is above a certain age. Therefore, as an alternative, we may consider a dynamic interventions that does not intervene on such community members, as determined by the pre-specified function $\delta(A_i, W_i)$. This in turn allows us to define the types of causal parameters which are less likely to violate the positivity assumption (also known as the assumption of experimental treatment assignment (ETA)).

In our simulation, we start by sampling a network graph for N units (nodes) according to the small world model [135], along with three unit-specific baseline covariates,

$$W_i = (W_i(1), W_i(2), W_i(3)),$$

for $i = 1, \ldots, N$. The unit-specific exposure A_i is then was simulated conditionally on W_i as normal with $\mu(W_i) = 0.58 * W_i(2) + 0.33 * W_i(3)$ and $\sigma^2 = 1$. We also let the conditional density of A_i given W_i to be denoted as $g_0(a|w)$. We denote the intervened exposure on i as A_i^*, with its corresponding conditional density denoted as g^*. Note that A_i^* is set equal to $A_i + shift$, for known constant $shift > 0$, only if the following condition holds:

$$\exp\{shift * (A_i - \mu(W_i) + shift/2)\} \geq trunc, \qquad (*)$$

for known truncation constant $trunc > 0$, and otherwise, the intervention keeps the observed exposure A_i unchanged. Finally, the binary outcome Y_i is simulated as dependent on all three of i's baseline covariates in W_i, i's exposure A_i, as well as the baseline covariate values and exposures of i's friends via the following network summaries: $1/|F_i| \sum_{j \in F_i} W_j(1)$ and $\sum_{j \in F_i} A_j$. To reiterate, we assume that the probability of success for each binary outcome Y_i is logit-linear function of the summary measures (A_i^s, W_i^s), which are defined as

$$A_i^s := (A_i, 1/|F_i| \sum_{j \in F_i} W_j(1)) \text{ and } W_i^s := (W_i, \sum_{j \in F_i} A_j).$$

Our target causal quantity is then defined with respect to N i-specific counterfactual outcomes Y_i^*, under stochastic intervention g^*, for $i = 1, \ldots, N$. These counterfactual outcomes are generated by replacing the structural equation for generating N exposures A_i with new structural equations for generating intervened exposures A_i^*, for $i = 1, \ldots, N$, and then sampling the outcomes Y_i^*, for $i = 1, \ldots, N$ from such a modified data-generating distribution. Note that we are using the new notation Y_i^* to denote the fact that the conditional distribution of the observed data (W_i, A_i, Y_i) has been modified by replacing g_0 with the intervention g^*. Our causal quantity is denoted by ψ_0 and we define it as the sample-average of the expected counterfactual outcomes, i.e.,

$$\psi_0 := 1/n \sum_{i=1}^{n} E[Y_i^*].$$

Using simcausal to specify the structural equation model for dependent data

In this simulation study we sample the network graph based on the small world network model [135], where its corresponding network generator function "`rnet.SmWorld`" is provided

in the **simcausal** package. The actual network sampling is performed by calling the function `sample_smallworld` of the **igraph** R package. The rest of the code simply converts the **igraph** output object into the **simcausal** network representation. We remind the reader that **simcausal** represents the network as a matrix with N rows, each row i representing the set of friends F_i (social connections of unit i), where each F_i may be padded with extra `NA`s to make sure all F_i are of the same length. As in the example in the previous section, the following code snippet instantiates an empty `DAG` object, then uses this `DAG` object to define a network called '`Net`', which is based on the network generator `rnet.SmWorld()`.

```
D <- DAG.empty()
D <- D + network("Net", netfun = "rnet.SmWorld", dim = 1, nei = 9, p = 0.1)
```

Next, we define the three IID baseline covariates, categorical (0-5) $W_i(1)$ (variable `W1`), binary $W_i(2)$ (variable `W2`) and binary $W_i(3)$ (variable `W3`), as shown below:

```
D <- D +
  node("W1", distr = "rcat.b0", probs = c(0.0494, 0.1823, 0.2806, 0.2680, 0.1651, 0.0546)) +
  node("W2", distr = "rbern", prob = plogis(-0.2 + W1/3)) +
  node("W3", distr = "rbern", prob = 0.6)
```

In our next step, we define the conditional distribution of a continuous exposure A_i (variable `A`) , sampled as normal, with mean given as a linear combination of $W_i(2)$ and $W_i(3)$ (variables `W2` and `W3`) and the standard deviation of 1.

```
D <- D + node("A.obs", distr = "rnorm", mean = 0.58 * W2 + 0.33 * W3, sd = 1)
```

Finally, we model the conditional probability of success for the binary outcome Y_i (variable `Y` below) as a logit-linear function of (W_i, A_i) and the earlier described summary measures (W_i^s, A_i^s), where the latter are based on the network of connections in F_i (friends of unit i). In more detail, for each unit i, W_i^s is defined as the mean of covariates $W_j(1)$ for all observations j that are friends with i ($j \in F_i$) and A_i^s is defined as the sum of the exposures of all friends of i, i.e., all A_j such that j is in F_i.

```
D <- D +
  node("A", distr = "rconst", const = A.obs) +
  node("Y", distr = "rbern",
    prob = plogis(+0.35*A +
                  +0.10*sum(A[[1:Kmax]]) +
                  -0.5*ifelse(nF > 0, sum(W1[[1:Kmax]])/nF, 0) +
                  -0.5*W1 - 0.58*W2 - 0.33*W3),
    replaceNAw0 = TRUE)
Dset <- set.DAG(D, n.test = 200)
```

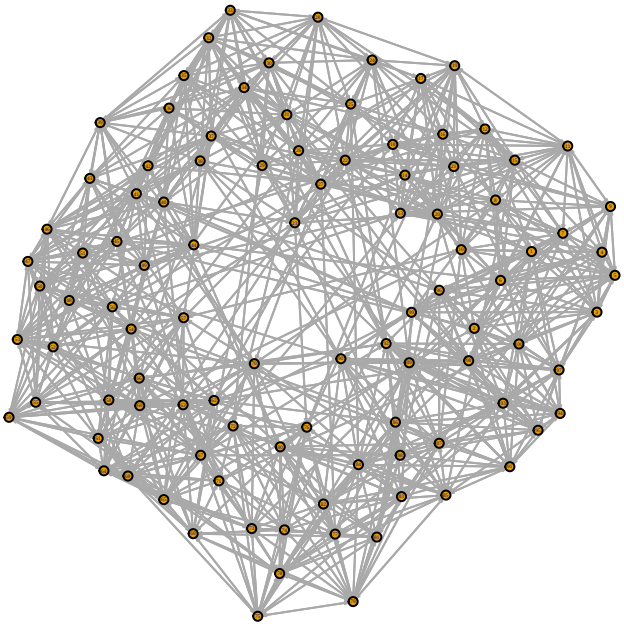

Figure 3.3: Network for 100 observations sampled according to the small world network model.

Note that the above object `Dset` has saved the described data-generating distribution, which includes the definition of the network and that of unit-specific data. In other words, `Dset` object saves all the information that is needed in order to be able to simulate: (1) the network graph on N units, and (2) the observations (W_i, A_i, Y_i), for $i = 1, \ldots, N$. Also note that such unit-specific data may or may not be dependent (i.e., it may or may not use the network structure), depending on a particular SEM parameterization selected by the user.

Simulating network and observed data

In our next code example we simulate the observed data on N units from the distribution specified in the above object `Dset`. As before, this is accomplished by calling the function `sim`, where the number of units is specified with the argument `n`. We save the resulting data frame on N observations (W_i, A_i, Y_i) as a variable `datO`, the network matrix as a variable `NetInd_mat` and the vector counting the number of friends for each unit as a variable `nF`.

```
nsamp <- 100
datO <- sim(Dset, n = nsamp, rndseed = 54321)
NetInd_mat <- attributes(datO)$netind_cl$NetInd
nF <- attributes(datO)$netind_cl$nF
```

As in the previous example, we use the `plot.igraph` function in `igraph` package to visualize the simulated network, as shown in Figure 3.3.

Defining interventions, simulating counterfactual data and evaluating the true value of the causal quantity

Our next goal is to evaluate the true value of the causal parameter ψ_0 under intervention g^* via Monte-Carlo simulation that samples from the distribution of the counterfactual outcomes Y_i^*, for $i = 1, \ldots, N$. This process consists of two stages. First, as we show in the following code snippet, we define the intervention of interest by modifying the observed data-generating distribution saved in the object `Dset`. Namely, we replace the conditional exposure density g_0 with a stochastic intervention given by g^*. Note that such interventions are defined by calling the function `action` and *adding* the result of this call to the previously defined object `Dset`. In the example given below, our intervention of interest is named '`gstar`' and it overrides the previous definition of the observed exposure node `A` with a new intervened definition that is based on the previously described stochastic intervention g^*.

```
Dset <- Dset +
  action("gstar",
    nodes = node("A", distr = "rconst",
      const = ifelse(exp(shift * (A.obs + shift -
                      (0.58*W2 + 0.33*W3) - shift/2)) > trunc,
                    A.obs,
                    A.obs + shift)),
    trunc = 1, shift = 0.5)
```

Second, we sample from such intervened (*post-intervention*) distribution defined by the action `gstar` and generate the counterfactual data, as shown in the following code example. We do this by calling the function `sim` and specifying the name of the previously defined action with an argument `actions`. The procedure in `sim` first samples the network of $N = 100,000$, proceeded by sampling the counterfactual data. We then evaluate the Monte-Carlo approximation of the true causal parameter by taking the mean of the simulated counterfactual outcomes Y_i^*, which provides a Monte-Carlo estimate of ψ_0. We note that this Monte-Carlo evaluated causal parameter defines our *gold standard*: the quantity which we may use for evaluating and comparing the performance of different statistical methods. One possible application of such causal parameters is exhibited in the following section.

```
nfull <- 100000
datFull <- sim(Dset, actions="gstar", n = nfull, rndseed = 54321)[[1]]
print(mean(datFull[["Y"]]))
```

3.6 Simulation study comparing performance of dependent-data estimators

We now describe one possible application for the **simcausal** package. In particular, we conduct a simulation study that compares the performance of the three dependent-data

estimators implemented in the **tmlenet** R package [116], based on the data generating distribution described in Section 3.5. We also note that the estimation algorithms implemented in the **tmlenet** package have been described elsewhere [114]. In particular, the **tmlenet** package implements three dependent-data estimators: Target Maximum Likelihood Estimator (TMLE), the Inverse Probability Weighted Estimator (IPW) and the G-computation estimator (GCOMP). We also note that our intervention of interest and the corresponding target causal quantity was introduced in the previous section. We will now use this target causal quantity as our *gold standard*, namely, it is the quantity that will generally remain known in a real data-generating process. Our simulation study hence uses this gold standard as a mean of comparing the finite sample performance of these three dependent-data estimators over iterated samples of hypothetical observed data. We also note that the inferential framework used by **tmlenet** assumes that the target causal quantity is defined conditionally on a particular sample size N and a particular network structure. Thus, the **simcausal** Monte-Carlo evaluation of our true causal quantity was slightly modified from the previous section as outlined below.

As our first step, we simulate the observed data, as shown below, for 10,000 units. In this example, we set the network-generating process to a fixed random seed and leave the rest of the data-generating procedure random.

```
datO <-  sim(Dset, n = 10000, rndseed = 544321, rndseed.reset.node = "W1")
net_obj <- attributes(datO)[["netind_cl"]]
NetInd_mat <- net_obj[["NetInd"]]
nF <- net_obj[["nF"]]
Kmax <- net_obj[["Kmax"]]
```

As our next step, we define the input parameters for the `tmlenet()` estimation function, as shown in the following code snippets. We define the observed-data baseline summary measures W_i^s by calling the function "`def_sW`" and the exposure summary measures A_i^s by calling the function "`def_sA`". Note, as we show below, these two functions use the syntax that is nearly identical to the **simcausal** syntax for specifying the network-based variable summaries.

```
require("tmlenet")
def_sW <- def_sW(W1, W2, W3) +
          def_sW(meanW1 = ifelse(nF > 0, sum(W1[[1:Kmax]])/nF, 0),
            replaceNAw0 = TRUE)
def_sA <- def_sA(A, sumA = sum(A[[1:Kmax]]), replaceNAw0 = TRUE)
```

In the following example we define the intervention of interest by calling the function "`def_new_sA`". We again note the syntactic similarities between the intervention specification in the example below and the specification of the counterfactual action `gstar` in the example from the previous section.

```
trunc <- 1; shift <- 0.5
newA.gstar <-  def_new_sA(A =
    ifelse(exp(shift * (A + shift - (0.58*W2 + 0.33*W3) - shift/2)) > trunc,
      A,
      A + shift))
```

In our following example we define the regression formulas, where the regression defined by the variable "`Qform`" is used for modeling the conditional outcome Y_i given the summary measures W_i^s, A_i^s and the regression in "`hform`" is used for modeling the conditional probabilities of the observed exposure summaries given the observed baseline summaries, i.e., $P(A_i^s|W_i^s)$. We also call `tmlenet_options` to specify additional tuning parameters for the **tmlenet** package. Finally, we call the estimation routine `tmlenet()`, as shown below.

```
Qform <- "Y ~ A + sumA + meanW1 + W1 + W2 + W3"
hform <- "A + sumA ~ meanW1 + W1 + W2 + W3"
tmlenet_options(bin.method = "equal.mass", maxNperBin = 200)
res <- tmlenet(data = datO, sW = def_sW, sA = def_sA,
                Kmax = Kmax, NETIDmat = NetInd_mat,
                intervene1.sA = newA.gstar ,
                Qform = Qform, hform.g0 = hform)
```

The above function call returns a list containing the three point estimates of the counterfactual sample-average expected outcome under intervention defined by `newA.gstar`, which can be obtained by running the following code (output not shown):

```
res[["EY_gstar1"]][["estimates"]]
```

The `tmlenet` function also returns the corresponding 95% confidence intervals (CIs) for TMLE and IPTW. Note that these CIs are adjusted for the dependence between units based on the input network structure and can be printed by running the following code (output not shown):

```
res[["EY_gstar1"]][["IC.CIs"]]
```

Finally, one can also obtain the 95% CIs which assume complete independence of the observed units, by running the following code (output not shown). One would expect that such independence-based CIs are going to be overly optimistic and will thus provide inadequate asymptotic coverage. The extent of this under-coverage is also explored in our simulation study.

```
res[["EY_gstar1"]][["iid.CIs"]]
```

Estimator	Bias10	MSE10	Variance10
TMLE	0.0081	0.00048	0.00048
IPTW	0.0037	0.00049	0.00049
GCOMP	0.0078	0.00041	0.00040

Table 3.1: Simulation-based performance of the three dependent data estimators across 1,000 simulations, each simulation consisted of $N = 10,000$ units. The reported bias, mean squared error (MSE) and variance are all multiplied by 10.

DEP.VAR	IID.VAR	Relative.VAR	DEP.CI.cover	IID.CI.cover
0.000048	0.000039	1.23	0.948	0.923

Table 3.2: Monte-Carlo approximated mean of the TMLE variance for dependent data ('DEP.VAR'), TMLE variance estimator for IID data ('IID.VAR'), the relative ratio of means of the two variances ('Relative.VAR'), the coverage of the 95% CI for dependent data ('DEP.CI.cover') and the coverage of the 95% CI for IID data ('IID.CI.cover').

Our simulation study repeated the above described estimation procedure 1,000 times, each time using a newly sampled dataset that used exactly the same network. The corresponding R code for this simulation study can be found in Appendix A. We evaluated the absolute bias, the mean-squared error (MSE) and the variance of these three estimators, with results presented in Table 3.1 (all performance metrics were multiplied by 10). In addition, we evaluated the mean of the TMLE variance estimate which adjusted for the dependence among units and the coverage of its corresponding 95% CI, as reported in columns "DEP.VAR" and "DEP.CI.cover" in Table 3.2. We also evaluated the mean of the TMLE variance estimate which assumed independence among units, along with its corresponding 95% CI, as reported in columns "IID.VAR" and "IID.CI.cover" of Table 3.2. Finally, we compared the relative ratio of the means of these two TMLE variance estimates, as reported in column "Relative.VAR" in Table 3.2. As expected, the IID-based variance estimator resulted in 95% CIs that had inadequate coverage, compared to the nearly nominal coverage probability of 0.95 for the TMLE variance estimator which appropriately adjusted for dependence among the units.

3.7 Extensions to longitudinal data structures

In our final example we demonstrate the extension of the **simcuasal** package towards simulations with longitudinal network data. This simulation is based on the longitudinal NPSEM for connected units and it has been previously described in [63]. We omit the description

of the technical details of such longitudinal NPSEMs in favor of an applied simulation example. We note that research on statistical methods for causal inference in longitudinal network data is a relatively new area of statistical research. Some of the recent packages that provide descriptive statistics and simulation for dynamic network data include **tsna** [10], **networkDynamic** [24], **RSiena** [99] and **netdiffuseR** [132].

In this simulation, we continue with an example first presented in Section 3.4 and demonstrate how it can be expanded towards simulations of dependent processes that also evolve in time over a static network. In particular, we present a simple model for the spread of an HIV epidemic over time based on a static network of 10 independent communities. We also note that another possible application of this type of a simulation study is for modeling the product or service adoption over time among the users of an online social network [34]. We assume that the study collects data on N individuals over time points $t = 0, \ldots, 50$, where for each unit i their network of sexual partners is denoted as F_i, their baseline covariates are denoted as W_i, and their indicator of receiving antiretroviral therapy is denoted as A_i. The unit's HIV status at time t is also recorded and is denoted as $Y_i(t)$, and the right-censoring indicator due to death is denoted as $D(t)$. We assume that whenever unit i is HIV-positive at t (i.e., $Y_i(t) = 1$), there is a non-zero probability of HIV transmission from i to another individuals j at time point $t+1$ if the unit i is in j's friend set (i.e., the event $Y_j(t+1) = 1$ can occur for all j such that $i \in F_j$). The same model also applies at next time cycle $t+1$, when a newly infected individual j can cause new HIV infections for all observations that had j in their friend set, even if some or all of those units were not at risk of getting HIV prior to cycle $t+2$. In this manner, any outcome $Y_j(t+2)$ measured at time $t+2$ is clearly dependent on the outcomes measured at $t+1$ for all units that were immediate sexual partners of j, i.e., all outcomes $(Y_k(t+1) : k \in F_j)$. However, the outcomes $(Y_k(t+1) : k \in F_j)$, and hence $Y_j(t+2)$, depend on an ever larger set of units as a function of outcomes measured at t, namely, all units within at most a second-degree of connectivity of j or the units in the set $\cup_{k \in F_i} F_k$ (friends of friends).

Our example below provides an intentionally simplified simulation of the above described process of contagion, where we use the time-varying Bernoulli node "`Y`" to model $Y(t)$, for $t = 0, \ldots, 50$. This simulation assumes that the conditional probability that unit i will be newly infected at each cycle t ($i.e., P(Y_i(t) = 1|\cdot)$), is a logit-linear function of the total number of friends of i who: (1) were infected at the previous time-point $t-1$ (all $j \in F_i$ such that $Y_j(t-1) = 1$); and (2) did not receive the antiretroviral treatment at baseline (all $j \in F_i$ with $A_j = 0$). In addition, we also model the right-censoring process $D(t)$, for $t = 0, ..., 50$ with a time-varying node "`D`", as shown below. Our network is defined by the number of independent clusters (communities), where the number of such clusters is determined by the variable "`nC`" and each cluster is inter-connected by a network sampled from the small world network model (the `R` code provided in Appendix B).

As before, we first add this new network to the `DAG` object `D1` previously defined in Section 3.4, naming it as `SmWrld.nC10`. Note that the variables that were already defined in the object `D1` will be based on the previously added network model, however, all of the new variables that we add next will be using this new network `SmWrld.nC10`. The distribution

of the exposure A (i.e., the indicator of receiving antiretroviral therapy) is unchanged from the simulation example in Section 3.4.

```
D1.l <- D1 +
  network("SmWrld.nC10", netfun = "rnet.SmWrld.nC", nC = 10, nei = 4, p = 0.05)
```

We are now ready to define the time-varying structure, using the same node function call, but with an additional argument t. We also define the summary measure netYA, which is equal the number of number of untreated and infected friends at a particular time point. We define the time-varying node "Y" as our outcome, where at each time-point t, the outcome node Y[t] is a logit-linear function of the summary "netYA[t]", a function of the exposure "A", and finally, a function of the baseline summary "sumW2". We also define a time-varying right-censoring node "D[t]" (death status), which depends on the baseline value of node "W1" and the HIV infection status at time t, namely, "Y[t]". The following example first defines the distributions of the time-varying nodes at their initial time-point $t = 0$.

```
D1.l <- D1.l +
  node("Y", t = 0, distr = "rbern",
    prob = ifelse(W3==1L, 1L,
      plogis(-4 - 2*sumW2 - 2*A + 1.5*sum(W3[[1:Kmax]]*(1-A[[1:Kmax]])))),
    replaceNAw0 = TRUE) +
  node("D", t = 0, distr = "rbern", prob = 0, EFU = TRUE)
```

The following code example defines the distributions of the time-varying nodes "netYA", "Y" and "D" for time-points $t = 1, \ldots, 50$:

```
D1.l <- D1.l +
  node("netYA", t = 1:50, distr = "rconst",
    const = sum(Y[t-1][[1:Kmax]]*(1-A[[1:Kmax]])), replaceNAw0 = TRUE) +
  node("Y", t = 1:50, distr = "rbern",
    prob = ifelse(Y[t-1]==1, 1, plogis(-4 - 5*sumW2 - 10*A + 0.5*netYA[t]))) +
  node("D", t = 1:50, distr = "rbern",
    prob = plogis(-7.5 + 0.25*W1  + 0.05*A+ 0.25*Y[t]), EFU = TRUE)
```

We finish defining the above data-generating distribution with a call to set.DAG function and we simulate a dataset with 10,000 observations, as shown below:

```
Dset <- set.DAG(D1.l, n.test = 1000,
  latent.v =c("sumW1", "sumW2", "sumW3", "W1.F1", "W1.F2",
              "W1.F3", "W1.F4", "W1.F5", "sumW3A"))
dat <- sim(Dset, n = 10000, LTCF = "D", rndseed = 543)
```

Below we present a sample of the simulated data for two observations and several time points. We also present the mean number of untreated and infected friends over time in a top plot of Figure 3.4 and we present the mortality and HIV prevalence over time in a bottom plot of Figure 3.4.

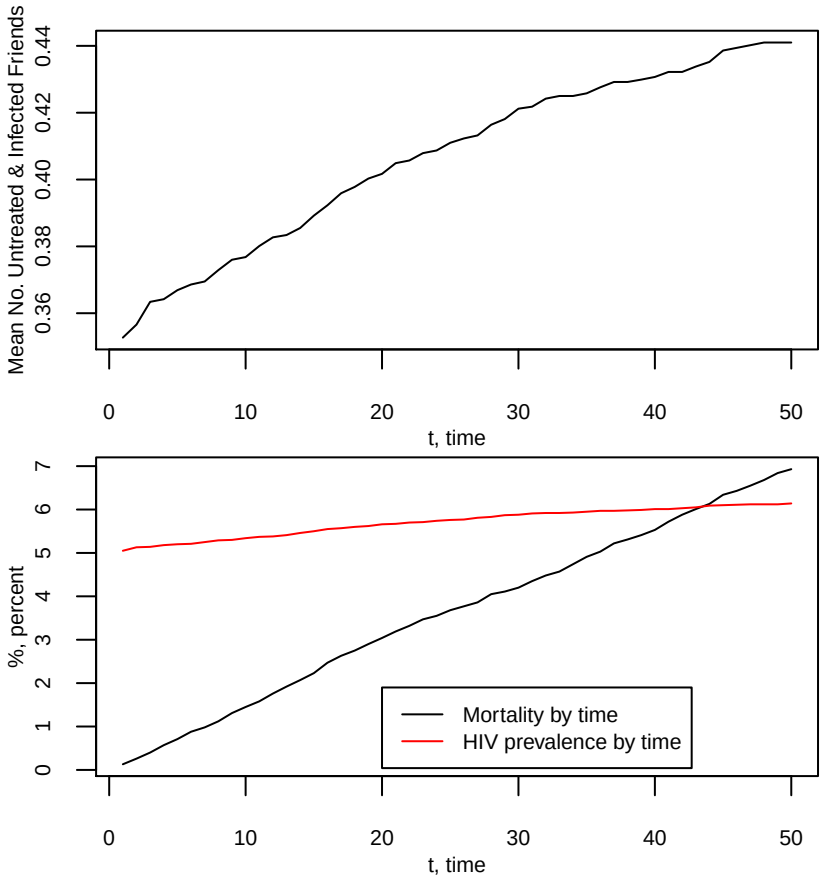

Figure 3.4: Mean number of untreated and infected friends by time (top figure). Mortality and HIV prevalence by time (bottom figure).

```
head(dat[c(1:2),1:19])
```

```
##   ID nF W1 W2 W3 A Y_0 D_0 netYA_1 Y_1 D_1 netYA_2 Y_2 D_2 netYA_3 Y_3 D_3
## 1  1  4  5  1  0 0   0   0       0   0   0       0   0   0       0   0   0
## 2  2  5  3  1  0 0   0   0       0   0   0       0   0   0       0   0   0
```

Suppose now that we also want to model the outcome $Y_i[t]$ as a function of another time-varying network summary defined for each unit i as the mean duration of HIV among all friends of observation i up to time t. Note that the duration of HIV for unit j at cycle t is defined by the total number of cycles since the initial HIV infection (first time Y_j jumps to 1). Below, we provide an example that defines exactly such a summary, called "`new.S`", by combining the time-indexing and network-indexing summaries into a single expression "`sum(sum(Y[0:(t)])[[0:Kmax]])/nF`". Note that this expression should be read from the inside out as follows: First, the expression "`sum(Y[0:(t)])`" evaluates to a sum of the unit-specific values of the time-varying vector `Y` from time-point 0 up to the current time-point value in `t`, i.e., the total number of cycles the person had HIV up to this point; Second, the

expression "`(...)[[0:Kmax]]`" evaluates the result of the previous expression for a network-based index "`[[0:Kmax]]`", which provides the total duration of HIV among all friends of each unit; Third, the expression "`sum(...)/nF`" is applied to the result of the previous expression to find the mean value (mean duration) of HIV among all friends of each unit.

```
D1.l <- D1.l +
    node("new.S", distr = "rconst", t = 1:50,
      const = sum(sum(Y[0:t])[[0:Kmax]])/nF, replaceNAw0 = TRUE)
```

Note that the above time-varying node "`new.S`" was added to the previously defined `DAG` object "`D1.l`". In particular, for a given time-point value `t` the node "`new.S[t]`" is added *after* the previously defined nodes "`netYA[t]`", "`Y[t]`" and "`D[t]`". The following example simulates the observed data with new summary node "`new.S`" and displays its value over time for three selected observations:

```
Dset.new <- set.DAG(D1.l, n.test = 1000, verbose = FALSE)
dat.new <- sim(Dset.new, n = 10000, rndseed = 757)
round(dat.new[c(1,5000, 8000), paste0("new.S_",1:10)],2)
```

```
##      new.S_1 new.S_2 new.S_3 new.S_4 new.S_5 new.S_6 new.S_7 new.S_8 new.S_9
## 1       0.29    0.43    0.57    0.71    0.86    1.00    1.14    1.29    1.43
## 5000    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
## 8000    0.00    0.00    0.00    0.00    0.00    0.14    0.29    0.43    0.57
```

3.8 Discussion

We described how the **simcausal** R package can facilitate the conduct of network-based simulation studies in causal inference research, specifically allowing one to model data with known network and known functional form of dependence among units. We also described how the package can be used for simulations with longitudinal data structures. We described how the **simcausal** R package allows creating a wide range of artificial datasets often encountered in public health applications of causal inference methods. In particular, this includes simulations of causal mechanisms of interference or spillover. To simplify the specification of such models we implemented a novel R syntax which allows for a concise and intuitive expression of complex functional dependencies for a large number of nodes. We also demonstrated how this syntax can be used for simplifying the specification and simulation of complex network-based summaries of the data. Moreover, we argued that such complex simulations are often necessary when one tries to conduct a realistic simulation study that attempts to replicate a large variety of scenarios one might expect to see from a true data-generating process. We also note that our package can work in conjunction with other network simulation tools, as we demonstrated with the example of the **igraph** R package that was used to sample the actual networks. In addition, the **simcausal** package allows the user to specify and simulate counterfactual data under various interventions (e.g., static, dynamic, deterministic, or

stochastic). These interventions may represent exposures to treatment regimens, the occurrence or non-occurrence of right-censoring events, or of specific monitoring events. These simulations provide practical settings with the types of data generating distributions which might be used for validation, testing and comparison of statistical methods for causal inference. To the best of our knowledge there are no other tools which implement the specific type of functionality afforded by the network-based syntax that is implemented in **simcausal** and is described in this chapter.

We note that one of the distinguishing features of **simcausal** is that it allows the user to define and compute various causal target parameters, such as, the treatment-specific counterfactual mean, that can then serve as the model-free *gold standard*. That is, the causal parameter is always the same functional of the counterfactual data distribution, regardless of the user-selected parameterization of the structural equation model. For example, the gold standard defined in this manner provides an objective measure of bias that does not depend on the modeling assumptions of a specific statistical method. Coupled with a wide variety of possible data generating distributions that may be specified in **simcausal**, this package provides statisticians with a powerful tool for testing the validity and accuracy of various statistical methods. For example, one may use our package for validating an implementation of a novel statistical method, using the simulated data with the known truth (the true value of the causal parameter), prior to applying such an algorithm to real data, in which this truth is unknown. As another example, one may use **simcausal** to simulate data from a variety of data-generating distributions and conduct a simulation study comparing the properties of different statistical procedures (e.g., bias, mean-squared error (MSE), asymptotic confidence interval coverage) against the user-selected causal parameter. We emphasize that the main purpose of our package is not to assess the impact of real-life interventions, but rather to test the validity and performance of statistical methods, which can then be applied to real datasets. Moreover, we demonstrated that the **simcausal** R package is a flexible tool that enables easier communication of assumptions between various practitioners and thus helps improve the transparency about the assumptions of different statistical methods.

We demonstrated the functionality of the **simcausal** package and its support for a large variety of network-based data generating processes with two single time point intervention simulation studies in Sections 3.4 and 3.5, as well the simulation study of more complex network-based processes that evolve in time in Section 3.7. We also showed a real-world application of the **simcausal** package in Section 3.6, where we conducted a simulation study that used another R package **tmlenet** to evaluate the performance of the three estimators of causal effect under interference intervening on the single time-point continuous exposure. We also note that the **simcausal** package vignette "*simcausal Package: Simulations with Complex Longitudinal Data*" [115] contains additional examples for simulations with IID data and other technical details of **simcausal** functionality. We also note that the implementation of additional features in future releases of the **simcausal** package should further expand its utility for methods research. Among such possible improvements is to allow interventions on the network structure and providing a unified interface for changes to friend structure as part of the intervention. Future work will also focus on modeling the changes in network structure

over time, for example, by providing an interface for specifying a time-varying analogs of the `network` function, modeling the probability of forming a new tie or removing a certain friend over time and allowing one to sample new networks conditional on the previously sampled networks.

3.9 Chapter apendix

R code for a network simulation study

```
require("doParallel")
registerDoParallel(cores = detectCores())
nsamp <- 10000
n.sim <- 1000
psi0.reps <- foreach(i.sim = seq(500), .combine = "c") %dopar% {
  datFull <- sim(Dset, actions="gstar", n = nsamp,
                 rndseed = 544321, rndseed.reset.node = "W1")[[1]]
  psi0 <- mean(datFull[["Y"]])
}
(psi0 <- mean(psi0.reps)) # 0.2297232

psi.n.mat <- foreach(i.sim = seq(n.sim), .combine = "rbind") %dopar% {
  dat0 <- sim(Dset, n = nsamp, rndseed = 544321, rndseed.reset.node = "W1")
  res <- tmlenet(data = dat0, sW = def_sW, sA = def_sA,
                Kmax = attributes(dat0)[["netind_cl"]][["Kmax"]],
                NETIDmat = attributes(dat0)[["netind_cl"]][["NetInd"]],
                intervene1.sA = newA.gstar,
                Qform = Qform, hform.g0 = hform)
  psi.n.vec <- t(res[["EY_gstar1"]][["estimates"]][,1])
  IID.tmle.CI <- res[["EY_gstar1"]][["iid.CIs"]]["tmle", ]
  tmle.CI <- res[["EY_gstar1"]][["IC.CIs"]]["tmle", ]
  IID.tmle.cover <- as.vector(IID.tmle.CI[1] <= psi0 & psi0 <= IID.tmle.CI[2])
  tmle.cover <- as.vector(tmle.CI[1] <= psi0 & psi0 <= tmle.CI[2])
  c(psi.n.vec,
    var.iid.TMLE = res[["EY_gstar1"]][["iid.vars"]]["tmle", ],
    var.TMLE = res[["EY_gstar1"]][["IC.vars"]]["tmle", ],
    IID.tmle.cover = IID.tmle.cover, tmle.cover = tmle.cover)
}
save(list=c("psi.n.mat", "psi0"), file="psi.n.mat.rda")
```

R code for additional network generators

Example of a small world network model with independent clusters

Below we provide the R code for a network sampling function used in the simulation example with longitudinal data described in Section 3.7. Specifically this function samples networks of independent communities, where each community network is distributed according to the small world network model and the number of independent communities is specified with the argument `nC`. An example graph of such a network for 1,000 observations is shown in Figure 3.5.

```
rnet.SmWrld.nC <- function(n, nC = 1, nei, p, ...) {
  sample_1Comm <- function(size) {
    g <- igraph::sample_smallworld(dim = 1, size = size, nei = nei, p = p,
                                   loops = FALSE, multiple = FALSE)
    g <- igraph::as.directed(g, mode = c("mutual"))
    NetInd_mat_1C <- sparseAdjMat.to.NetInd(igraph.to.sparseAdjMat(g))
    return(NetInd_mat_1C[["NetInd_k"]])
  }

  padcolsNetMat <- function(NetInd_mat) {
    if (ncol(NetInd_mat)<maxF) {
      NetInd_mat <- cbind(NetInd_mat,
        matrix(NA,nrow=nrow(NetInd_mat),ncol=maxF-ncol(NetInd_mat)))
    }
    return(NetInd_mat)
  }

  if (n <= 100) {nC <- 1}
  size_C <- vector(mode="integer", length=nC)
  for (i in 1:nC) size_C[i] <- as.integer(n / nC)
  size_C[nC] <- size_C[nC] + (n-sum(size_C))
  stopifnot(sum(size_C)==n)
  NetInd_mat_list <- lapply(seq(nC), function(iC)
                       sample_1Comm(size_C[iC]) + (iC - 1)*size_C[1])
  maxF <- max(unlist((lapply(NetInd_mat_list, ncol))))
  NetInd_mat_list <- lapply(NetInd_mat_list, padcolsNetMat)
  NetInd_mat <- do.call('rbind', NetInd_mat_list)
  return(NetInd_mat)
}
```

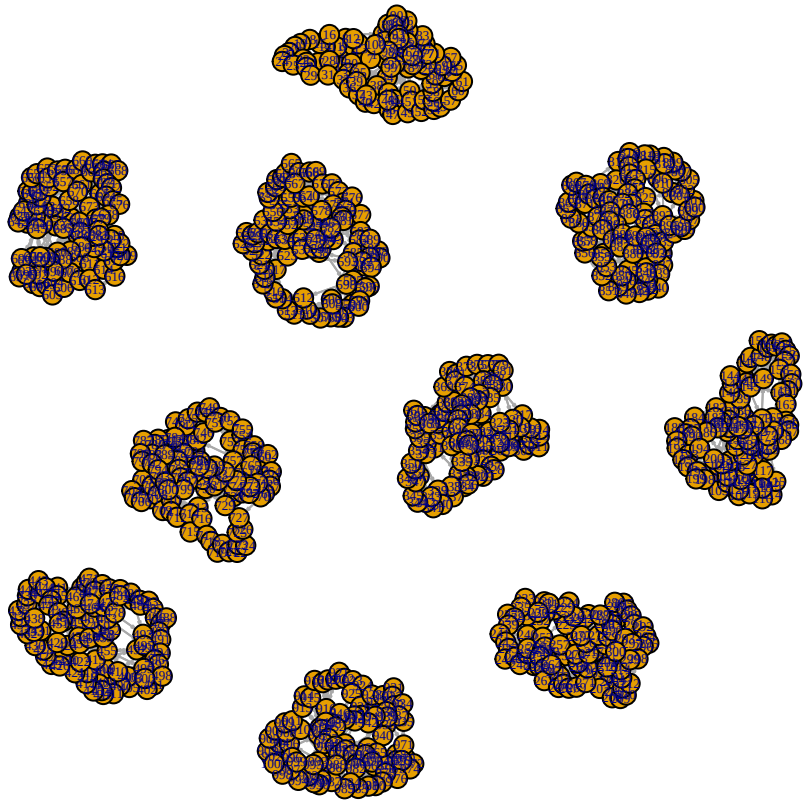

Figure 3.5: Network for 100 observations sampled by the custom network generator.

Example of a custom network generator

In this example we demonstrate that the user can define their own custom network sampling function, such as the `genNET` function provided below. Specifically, we will sample a network for `n` units, where the probability of making a connection (friendship) between two units is defined conditionally on the units' own baseline covariates. The following arguments are passed to the function `genNET()`:

- `maxFi` - The overall maximal number of friends
- `bslVar` - The baseline covariate which is used for constructing weights for the probability of selecting unit i as someone else's friend (weighted sampling);
- `nF` - A vector with the total number of friends that needs to be sampled for each unit i.

```
genNET <- function(n, maxFi, bslVar, nF, ...) {
  prob_F <- plogis(-4.5 + 2.5*c(1:6)/2) / sum(plogis(-4.5 + 2.5*c(1:6)/2))
  NetInd_k <- matrix(NA_integer_, nrow = n, ncol = maxFi)
  nFriendTot <- rep(0L, n)
  for (index in (1:n)) {
```

```
    FriendSampSet <- setdiff(c(1:n), index)
    nFriendSamp <- max(nF[index] - nFriendTot[index], 0L)
    if (nFriendSamp > 0) {
      if (length(FriendSampSet) == 1)  {
        friends_i <- FriendSampSet
      } else {
        friends_i <- sort(sample(FriendSampSet, size = nFriendSamp,
                          prob = prob_F[bslVar[FriendSampSet] + 1]))
      }
      NetInd_k[index, ] <- c(as.integer(friends_i),
                             rep_len(NA_integer_, maxFi - length(friends_i)))
      nFriendTot[index] <- nFriendTot[index] + nFriendSamp
    }
  }
  return(NetInd_k)
}
```

Next, we define the matrix of categorical probabilities `p.nFbyW1`. We will use this matrix for sampling the total number of friends (`nF`), conditionally on the value of the baseline covariate `W1`. Specifically, if `W1` is equal to 0, the observation's number of friends, 0 to 6, is sampled according to the probabilities in the first column of the matrix `p.nFbyW1`, and so on, each observation-specific value of `W1` being used to look up observation-specific categorical probabilities from `p.nFbyW1`.

```
set.seed(544321)
normprob <- function(x) x / sum(x)
p.nFbyW1 <- apply(matrix(runif(7*6), ncol = 6, nrow = 7), 2, normprob)
colnames(p.nFbyW1) <- paste0("p.nFbyW1_", c(0:5))
print(p.nFbyW1[c(1:2),])
```

```
##      p.nFbyW1_0 p.nFbyW1_1 p.nFbyW1_2 p.nFbyW1_3 p.nFbyW1_4 p.nFbyW1_5
## [1,]  0.1461838  0.1526971  0.1006473  0.1240505 0.02911188  0.2123011
## [2,]  0.1738377  0.1833152  0.2790983  0.1569190 0.23426583  0.2273970
```

In this simulation, the baseline covariate `W1` is added to the `DAG` object prior to registering the network generator. The network will be sampled later and the sampling process will be performed conditional on the value of the categorical `W1`, which is passed as an argument `bslVar` to the network generator function `genNET()`.

```
create_probs_nF <- function(W1) t(p.nFbyW1[,W1+1])
vecfun.add("create_probs_nF")
```

```
D <- DAG.empty()
D <- D +
  node("W1", distr = "rcat.b0",
    probs = c(0.0494, 0.1823, 0.2806, 0.2680, 0.1651, 0.0546))
```

Next, we define the node for the total number of friends (nF) according to the rule described above. We pass W1 to function create_probs_nF[7], which then looks-up the unit-specific vector of categorical probabilities from matrix p.nFbyW1, depending on the specific value of W1.

```
D <- D +
  node("nF", distr = "rcat.b0", probs = create_probs_nF(W1)) +
  network(name = "net.custom", netfun = "genNET",
    maxFi = 6, bslVar = W1, nF = nF)
  Dset <- set.DAG(D, n.test = 200)
```

We then simulate 100 observations by calling the function sim, as shown below.

```
nsamp <- 100
dat0 <- sim(Dset, n = nsamp, rndseed = 54321)
NetInd_mat <- attributes(dat0)$netind_cl$NetInd
nF <- attributes(dat0)$netind_cl$nF
```

We use the plot.igraph function in igraph package to visualize the simulated network, as shown in Figure 3.6.

[7]Note that prior to defining a categorical node "nF", we called vecfun.add(...), to define the vectorized function create_probs_nF. See the package vignette "*simcausal Package: Simulations with Complex Longitudinal Data*", Section 3.6, Subsection *Vectorizing node formula functions* for the description of why this needs to be done. Next, we add the network generator genNET to D, passing it the above defined observation-specific variables, i.e., the total number of friends nF and the categorical W1.

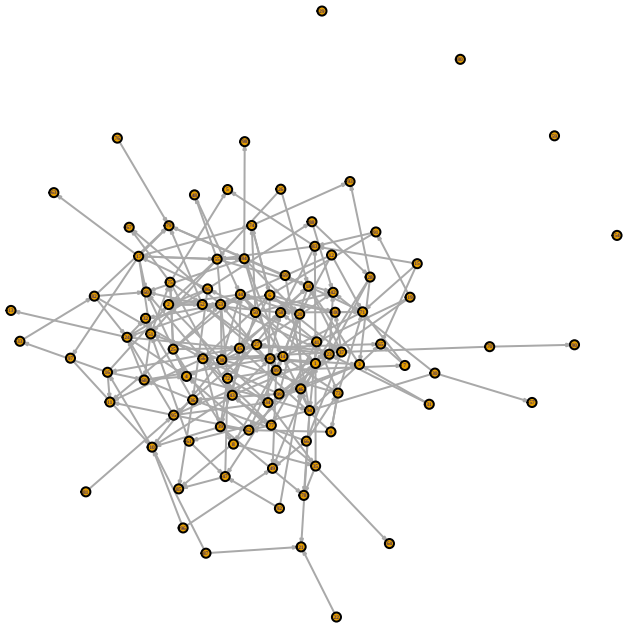

Figure 3.6: Network for 100 observations sampled by the custom network generator.

Bibliography

[1] Samrachana Adhikari et al. *CIDnetworks: Generative Models for Complex Networks with Conditionally Independent Dyadic Structure*. R package version 0.8.1. 2015. URL: https://CRAN.R-project.org/package=CIDnetworks.

[2] Sinan Aral and Dylan Walker. "Identifying Social Influence in Networks Using Randomized Experiments". In: *IEEE Intelligent Systems* 26.5 (2011), pp. 91–96. DOI: 10.1109/MIS.2011.89.

[3] Sinan Aral and Dylan Walker. "Tie Strength, Embeddedness, and Social Influence: A Large-Scale Networked Experiment". In: *Management Science* 60.6 (2014), pp. 1352–1370. DOI: 10.1287/mnsc.2014.1936.

[4] Peter M. Aronow and Cyrus Samii. "Estimating Average Causal Effects Under Interference Between Units". In: *ArXiv e-prints* (May 2013). arXiv: 1305.6156.

[5] Susan Athey, Dean Eckles, and Guido W. Imbens. *Exact P-values for Network Interference*. Working Paper 21313. National Bureau of Economic Research, July 2015. DOI: 10.3386/w21313.

[6] Laura Balzer, Maya Petersen, and Mark van der Laan. "Targeted Estimation and Inference for the Sample Average Treatment Effect". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 334 (2015).

[7] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pp. 509–512.

[8] Elias Bareinboim and Judea Pearl. "A general algorithm for deciding transportability of experimental results". In: *Journal of Causal Inference* 1.1 (2013), pp. 107–134.

[9] Guillaume W. Basse and Edoardo M. Airoldi. "Optimal design of experiments in the presence of network-correlated outcomes". In: *ArXiv e-prints* (July 2015). arXiv: 1507.00803.

[10] Skye Bender-deMoll and Martina Morris. *tsna: Tools for Temporal Social Network Analysis*. R package version 0.2.0. 2016. URL: https://CRAN.R-project.org/package=tsna.

[11] Christoph Bergmeir and José M. Benítez. *RSNNS: Neural Networks in R using the Stuttgart Neural Network Simulator (SNNS)*. R package version 0.4-7. 2015. URL: https://CRAN.R-project.org/package=RSNNS.

[12] Peter J Bickel. *Efficient and adaptive estimation for semiparametric models*. New York: Springer-Verlag., 1993, pp. xix, 560. ISBN: 0801845416 (acid-free paper).

[13] Nello Blaser et al. "gems: An R Package for Simulating from Disease Progression Models". In: *Journal of Statistical Software* 64.10 (2015), pp. 1–22. URL: http://www.jstatsoft.org/v64/i10/.

[14] Steven M. Boker et al. *OpenMx: Multipurpose Software for Statistical Modeling*. R package version 2.0.1-4157. 2014. URL: http://openmx.psyc.virginia.edu.

[15] Steven Boker et al. "OpenMx: an open source extended structural equation modeling framework". In: *Psychometrika* 76.2 (2011), pp. 306–317.

[16] Kenneth A. Bollen. *Structural Equations with Latent Variables*. John Wiley & Sons, 1989.

[17] Kenneth A Bollen and Judea Pearl. "Eight Myths About Causality and Structural Equation Models". In: *Handbook of Causal Analysis for Social Research*. Springer, 2013, pp. 301–328.

[18] Jake Bowers, Mark M. Fredrickson, and Costas Panagopoulos. "Reasoning about interference between units: A general framework". In: *Political Analysis* 21.1 (2013), pp. 97–124. ISSN: 10471987. DOI: 10.1093/pan/mps038.

[19] Francois Briatte. *ggnetwork: Geometries to Plot Networks with 'ggplot2'*. R package version 0.5.1. 2016. URL: https://CRAN.R-project.org/package=ggnetwork.

[20] M. Alan Brookhart et al. "Variable Selection for Propensity Score Models". In: *American Journal of Epidemiology* 163.12 (2006), pp. 1149–1156.

[21] Christian Brownlees. *nets: Network Estimation for Time Series*. R package version 0.8. 2016. URL: https://CRAN.R-project.org/package=nets.

[22] Andrea Burton et al. "The Design of Simulation Studies in Medical Statistics". In: *Statistics in Medicine* 25.24 (2006), pp. 4279–4292.

[23] Carter T. Butts. *sna: Tools for Social Network Analysis*. R package version 2.3-2. 2014. URL: https://CRAN.R-project.org/package=sna.

[24] Carter T. Butts et al. *networkDynamic: Dynamic Extensions for Network Objects*. R package version 0.9.0. 2016. URL: https://CRAN.R-project.org/package=networkDynamic.

[25] David S. Choi. "Estimation of Monotone Treatment Effects in Network Experiments". In: *ArXiv e-prints* (Aug. 2014). arXiv: 1408.4102.

[26] Nicholas A Christakis and James H Fowler. "Social contagion theory: examining dynamic social networks and human behavior". In: *Statistics in Medicine* 32.4 (2013), pp. 556–577.

[27] Nicholas A Christakis and James H Fowler. "The spread of obesity in a large social network over 32 years". In: *New England Journal of Medicine* 357.4 (2007), pp. 370–379.

[28] L. M. Collins, J. L. Schafer, and C. M. Kam. "A Comparison of Inclusive and Restrictive Strategies in Modern Missing Data Procedures". In: *Psychological Methods* 6.4 (Dec. 2001), pp. 330–351.

[29] G Csardi and T Nepusz. "The igraph software package for complex network research". In: *InterJournal* Complex Systems (2006), p. 1695. DOI: 10.1109/ICCSN.2010.34. URL: http://igraph.org.

[30] Calum Davey et al. "Re-analysis of health and educational impacts of a school-based deworming programme in western Kenya: a statistical replication of a cluster quasi-randomized stepped-wedge trial". In: *International Journal of Epidemiology* (2015). DOI: 10.1093/ije/dyv128.

[31] A Philip Dawid, Vanessa Didelez, et al. "Identifying the consequences of dynamic treatment strategies: A decision-theoretic overview". In: *Statistics Surveys* 4 (2010), pp. 184–231.

[32] Hakan Demirtas. "The Design of Simulation Studies in Medical Statistics by Andrea Burton, Douglas G. Altman, Patrick Royston and Roger L. Holder, Statistics in Medicine 2006; 25:4279-4292". In: *Statistics in Medicine* 26.20 (2007), pp. 3818–3821.

[33] M Dowle et al. *data.table: Extension of data.frame*. R package version 1.9.4. 2014. URL: http://CRAN.R-project.org/package=data.table.

[34] Dean Eckles, Brian Karrer, and Johan Ugander. "Design and Analysis of Experiments in Networks: Reducing Bias from Interference". In: *arXiv preprint arXiv:1404.7530* (2014).

[35] Felix Elwert. "Graphical Causal Models". In: *Handbook of Causal Analysis for Social Research*. Springer, 2013, pp. 245–273.

[36] Felix Elwert and Christopher Winship. "Endogenous Selection Bias: The Problem of Conditioning on a Collider Variable". In: *Annual Review of Sociology* 40 (2014), pp. 31–53.

[37] Zoe Fewell, George Davey Smith, and Jonathan A. C. Sterne. "The Impact of Residual and Unmeasured Confounding in Epidemiologic Studies: A Simulation Study". In: *American Journal of Epidemiology* 166.6 (2007), pp. 646–655.

[38] John Fox. "Teacher's corner: structural equation modeling with the sem package in R". In: *Structural equation modeling* 13.3 (2006), pp. 465–486.

[39] John Fox, Zhenghua Nie, and Jarrett Byrnes. *sem: Structural Equation Models*. R package version 3.1-5. 2014. URL: http://CRAN.R-project.org/package=sem.

[40] Christopher Gandrud. *d3Network: Tools for creating D3 JavaScript network, tree, dendrogram, and Sankey graphs from R*. R package version 0.5.2.1. 2015. URL: https://CRAN.R-project.org/package=d3Network.

[41] Richard D Gill and James M Robins. "Causal inference for complex longitudinal data: the continuous case". In: *Annals of Statistics* (2001), pp. 1785–1811.

[42] Ryan Gill, Somnath Datta, and Susmita Datta. *dna: Differential Network Analysis*. R package version 1.1-1. 2014. URL: https://CRAN.R-project.org/package=dna.

[43] Adam Glynn and Kevin Quinn. *Non-parametric Mechanisms and Causal Modeling*. Tech. rep. 2007.

[44] Isabella Gollini. *lvm4net: Latent Variable Models for Networks*. R package version 0.2. 2015. URL: https://CRAN.R-project.org/package=lvm4net.

[45] Ravi Goyal, Victor De Gruttola, and Joseph Blitzstein. "Sampling Networks from Their Posterior Predictive Distribution". In: *Network Science (Cambridge University Press)* 2.1 (2014), pp. 107–131. ISSN: 2050-1242. DOI: 10.1017/nws.2014.2.

[46] Timothy Graham, Robert Ackland, and with contribution from Chung-hong Chan. *SocialMediaLab: Tools for Collecting Social Media Data and Generating Networks for Analysis*. R package version 0.22.0. 2016. URL: https://CRAN.R-project.org/package=SocialMediaLab.

[47] Tom Greene et al. "The Balanced Survivor Average Causal Effect". In: *The International Journal of Biostatistics* 9.2 (2013), pp. 291–306.

[48] Chris Groendyke and David Welch. *epinet: Epidemic/Network-Related Tools*. R package version 2.1.7. 2016. URL: https://CRAN.R-project.org/package=epinet.

[49] Susan Gruber and Mark J van der Laan. "Targeted maximum likelihood estimation: A gentle introduction". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 252 (2009).

[50] Danella M Hafeman and Tyler J VanderWeele. "Alternative Assumptions for the Identification of Direct and Indirect Effects". In: *Epidemiology* 22.6 (2011), pp. 753–764.

[51] Mark S. Handcock et al. *ergm: Fit, Simulate and Diagnose Exponential-Family Models for Networks*. R package version 3.6.0. 2016. URL: https://CRAN.R-project.org/package=ergm.

[52] Mark S. Handcock et al. "statnet: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data". In: *Journal of Statistical Software* 24.1 (2008), pp. 1–11. URL: http://www.jstatsoft.org/v24/i01.

[53] Guy Harling et al. "Leveraging Contact Network Structure in the Design of Cluster Randomized Trials". In: *Harvard University Biostatistics Working Paper Series* Working Paper 199 (2016).

[54] Jennifer Hill and Jerome P. Reiter. "Interval Estimation for Treatment Effects Using Propensity Score Matching". In: *Statistics in Medicine* 25.13 (2006), pp. 2230–2256. ISSN: 1097-0258.

[55] Ted Hodgson and Maurice Burke. "On Simulation and the Teaching of Statistics". In: *Teaching Statistics* 22.3 (2000), pp. 91–96.

[56] Paul W Holland. "Statistics and Causal Inference". In: *Journal of the American Statistical Association* 81.396 (1986), pp. 945–960.

[57] Michael G Hudgens and M Elizabeth Halloran. "Toward causal inference with interference". In: *Journal of the American Statistical Association* 103.482 (2008).

[58] Guido W Imbens. "Nonparametric Estimation of Average Treatment Effects under Exogeneity: A Review". In: *Review of Economics and Statistics* 86.1 (2004), pp. 4–29.

[59] Samuel Jenness, Steven M. Goodreau, and Martina Morris. *EpiModel: Mathematical Modeling of Infectious Disease*. R package version 1.2.5. 2016. URL: `https://CRAN.R-project.org/package=EpiModel`.

[60] Marshall M Joffe, Dylan Small, Chi-Yuan Hsu, et al. "Defining and Estimating Intervention Effects for Groups That Will Develop an Auxiliary Outcome". In: *Statistical Science* 22.1 (2007), pp. 74–97.

[61] V. Kristman, M. Manno, and P. Cote. "Loss to Follow-Up in Cohort Studies: How Much is Too Much?" In: *European Journal of Epidemiology* 19.8 (2004), pp. 751–760.

[62] Pavel N. Krivitsky and Mark S. Handcock. *tergm: Fit, Simulate and Diagnose Models for Network Evolution Based on Exponential-Family Random Graph Models*. R package version 3.4.0. 2016. URL: `https://CRAN.R-project.org/package=tergm`.

[63] Mark J. van der Laan. "Causal Inference for a Population of Causally Connected Units". In: *Journal of Causal Inference* 2.1 (Mar. 2014), pp. 1–62. ISSN: 2193-3677. DOI: `10.1515/jci-2013-0002`.

[64] Mark J van der Laan. "Causal Inference for Networks". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 300 (2012).

[65] Mark J van der Laan, Alan E Hubbard, and Sara Kherad Pajouh. "Statistical inference for data adaptive target parameters". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 314 (2013).

[66] Mark J. van der Laan and Sherri Rose. *Targeted Learning: Causal Inference for Observational and Experimental Data*. Springer Series in Statistics. Springer-Verlag, 2011. ISBN: 978-1-4419-9781-4. DOI: `10.1007/978-1-4419-9782-1`.

[67] Mark J. van der Laan and Daniel Rubin. "Targeted Maximum Likelihood Learning". In: *The International Journal of Biostatistics* 2.1 (2006). ISSN: 1557-4679. DOI: `10.2202/1557-4679.1043`.

[68] G Lefebvre, J A Delaney, and R W Platt. "Impact of Mis-Specification of the Treatment Model on Estimates from a Marginal Structural Model". In: *Statistics in Medicine* 27.18 (2008), pp. 3629–3642.

[69] Lan Liu and Michael G Hudgens. "Large sample randomization inference of causal effects in the presence of interference." In: *Journal of the American Statistical Association* 109.505 (2014), pp. 288–301. ISSN: 0162-1459. DOI: 10.1080/01621459.2013.844698.

[70] Russell Lyons. "The Spread of Evidence-Poor Medicine via Flawed Social-Network Analysis". In: *Statistics, Politics, and Policy* 2.1 (2010), pp. 1–26. ISSN: 2151-7509. DOI: 10.2202/2151-7509.1024. arXiv: 1007.2876.

[71] Ross L Matsueda. "Key Advances in the History of Structural Equation Modeling". In: *Handbook of Structural Equation Modeling*. Ed. by Rick Hoyle. Guilford, New York, 2012.

[72] Edward Miguel and Michael Kremer. "Worms: Identifying Impacts on Education and Health in the Presence of Treatment Externalities". In: *Econometrica* 72.1 (2004), pp. 159–217. ISSN: 1468-0262. DOI: 10.1111/j.1468-0262.2004.00481.x.

[73] Armin Monecke and Friedrich Leisch. *semPLS: Structural Equation Modeling Using Partial Least Squares*. R package version 1.0-10. 2013. URL: http://CRAN.R-project.org/package=semPLS.

[74] David Moriña and Albert Navarro. *survsim: Simulation of Simple and Complex Survival Data*. R package version 1.1.3. 2015. URL: http://CRAN.R-project.org/package=survsim.

[75] Iván Díaz Muñoz and Mark van der Laan. "Population intervention causal effects based on stochastic interventions". In: *Biometrics* 68.2 (2012), pp. 541–549.

[76] Kairat Mynbaev and Carlos Martins-Filho. "Consistency and Asymptotic Normality for a Nonparametric Prediction under Measurement Errors". In: *Journal of Multivariate Analysis* 139 (2015), pp. 166–188.

[77] Romain Neugebauer and Mark van der Laan. "Nonparametric Causal Effects Based on Marginal Structural Models". In: *Journal of Statistical Planning and Inference* 137.2 (2007), pp. 419–434.

[78] Romain Neugebauer, Julie A. Schmittdiel, and Mark J. van der Laan. "Targeted Learning in Real-World Comparative Effectiveness Research with Time-Varying Interventions". In: *Statistics in Medicine* 33.14 (2014), pp. 2480–2520.

[79] Romain Neugebauer et al. "High-Dimensional Propensity Score Algorithm in Comparative Effectiveness Research with Time-Varying Interventions". In: *Statistics in Medicine* 34.5 (2015), pp. 753–781.

[80] Jerzy Neyman. "Sur les applications de la theorie des probabilites aux experiences agricoles: Essai des principes (In Polish). English translation by DM Dabrowska and TP Speed (1990)". In: *Statistical Science* 5 (1923), pp. 465–480.

[81] Hans Noel and Brendan Nyhan. "The "unfriending" problem: The consequences of homophily in friendship retention for causal estimates of social influence". In: *Social Networks* 33.3 (2011), pp. 211–218. ISSN: 0378-8733. DOI: http://dx.doi.org/10.1016/j.socnet.2011.05.003.

[82] Daniel Oberski. *lavaan.survey: Complex survey Structural Equation Modeling (SEM)*. R package version 1.1. 2014. URL: http://CRAN.R-project.org/package=lavaan.survey.

[83] E. L. Ogburn and T. J. VanderWeele. "Vaccines, Contagion, and Social Networks". In: *ArXiv e-prints* (Mar. 2014). arXiv: 1403.1241 [stat.ME].

[84] Judea Pearl. "An Introduction to Causal Inference". In: *The International Journal of Biostatistics* 6.2 (2010).

[85] Judea Pearl. "Causal Diagrams for Empirical Research". In: *Biometrika* 82.4 (1995), pp. 669–688.

[86] Judea Pearl. *Causality: Models, Reasoning and Inference*. 2nd. New York, NY, USA: Cambridge University Press, 2009. ISBN: 052189560X, 9780521895606.

[87] Judea Pearl. "Direct and Indirect Effects". In: *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. UAI'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 411–420.

[88] Judea Pearl. "The Causal Foundations of Structural Equation Modeling". In: *Handbook of Structural Equation Modeling*. Ed. by Rick Hoyle. Guilford, New York, 2012.

[89] Judea Pearl. "The Foundations of Causal Inference". In: *Sociological Methodology* 40.1 (2010), pp. 75–149.

[90] Judea Pearl and Elias Bareinboim. "External validity: From do-calculus to transportability across populations". In: *Statistical Science* 29.4 (2014), pp. 579–595.

[91] Judea Pearl and Elias Bareinboim. "Transportability of causal and statistical relations: A formal approach". In: *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)* (2011), pp. 247–254.

[92] Maya L Petersen, Sandra E Sinisi, and Mark J van der Laan. "Estimation of Direct Causal Effects". In: *Epidemiology* 17.3 (2006), pp. 276–284.

[93] Maya L Petersen et al. "Diagnosing and Responding to Violations in the Positivity Assumption". In: *Statistical Methods in Medical Research* 21.1 (2012), pp. 31–54.

[94] Sunthud Pornprasertmanit, Patrick Miller, and Alexander Schoemann. *simsem: SIMulated Structural Equation Modeling*. R package version 0.5-11. 2015. URL: http://CRAN.R-project.org/package=simsem.

[95] K. E. Porter et al. "The Relative Performance of Targeted Maximum Likelihood Estimators". In: *The International Journal of Biostatistics* 7.1 (2011), pp. 1–34.

[96] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2016. URL: https://www.R-project.org/.

[97] Andrea Rau. *ebdbNet: Empirical Bayes Estimation of Dynamic Bayesian Networks*. R package version 1.2.3. 2013. URL: https://CRAN.R-project.org/package=ebdbNet.

[98] A Renyi and P Erdos. "On Random Graphs". In: *Publicationes Mathematicae* 6.290-297 (1959), p. 5.

[99] Ruth Ripley, Krists Boitmanis, and Tom A.B. Snijders. *RSiena: Siena - Simulation Investigation for Empirical Network Analysis*. R package version 1.1-232. 2013. URL: https://CRAN.R-project.org/package=RSiena.

[100] Antonio Rivero Ostoic. *multiplex: Algebraic Tools for the Analysis of Multiple Social Networks*. R package version 1.8.2. 2016. URL: https://CRAN.R-project.org/package=multiplex.

[101] J M Robins. "Marginal Structural Models". In: *1997 proceedings of the American Statistical Association, section on Bayesian statistical science* (1998), pp. 1–10. URL: http://www.biostat.harvard.edu/~robins/research.html.

[102] James M Robins. "A graphical approach to the identification and estimation of causal parameters in mortality studies with sustained exposure periods." In: *Journal of Chronic Diseases* 40 Suppl 2 (1987), 139S–161S. ISSN: 00219681. DOI: 10.1016/S0021-9681(87)80018-8.

[103] James M Robins. "Addendum to "a new approach to causal inference in mortality studies with a sustained exposure period-application to control of the healthy worker survivor effect"". In: *Computers & Mathematics with Applications* 14.9-12 (1987), pp. 923–945. ISSN: 08981221. DOI: 10.1016/0898-1221(87)90238-0.

[104] James M Robins. "Causal inference from complex longitudinal data". In: *Latent variable modeling and applications to causality*. Springer, 1997, pp. 69–117. ISBN: 978-1-4612-1842-5. DOI: 10.1007/978-1-4612-1842-5_4. arXiv: NewYork,NY:Springer.

[105] James M Robins. "[choice as an alternative to control in observational studies]: comment". In: *Statistical Science* (1999), pp. 281–293.

[106] James M Robins and Thomas Richardson. "Alternative graphical causal models and the identification of direct effects". In: *Causality and Psychopathology: Finding the Determinants of Disorders and Their Cures* (2010), pp. 103–158.

[107] Paul R Rosenbaum. "Interference Between Units in Randomized Experiments". In: *Journal of the American Statistical Association* 102.477 (2007), pp. 191–200. ISSN: 0162-1459. DOI: 10.1198/016214506000001112.

[108] Yves Rosseel. *lavaan: Latent Variable Analysis*. R package version 0.5-18. 2015. URL: http://CRAN.R-project.org/package=lavaan.

[109] Fernando S. Marques, Jose H. H. Grisi-Filho, and Marcos Amaku. *hybridModels: Stochastic Hybrid Models in Dynamic Networks*. R package version 0.2.6. 2016. URL: https://CRAN.R-project.org/package=hybridModels.

[110] A. Sciandra, F. Gioachin, and L. Finos. *egonet: Tool for ego-centric measures in Social Network Analysis*. R package version 1.2. 2012. URL: https://CRAN.R-project.org/package=egonet.

[111] Cosma Rohilla Shalizi and Andrew C Thomas. "Homophily and contagion are generically confounded in observational social network studies". In: *Sociological methods & research* 40.2 (2011), pp. 211–239.

[112] I Shpitser and J Pearl. "Effects of Treatment on the Treated: Identification and Generalization". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. Montreal, Quebec: AUAI Press, 2009, pp. 514–521.

[113] Michael E Sobel. "What Do Randomized Studies of Housing Mobility Demonstrate?" In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1398–1407. DOI: 10.1198/016214506000000636.

[114] Oleg Sofrygin and Mark J van der Laan. "Semi-Parametric Estimation and Inference for the Mean Outcome of the Single Time-Point Intervention in a Causally Connected Population". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 344 (2015).

[115] Oleg Sofrygin, Mark J. van der Laan, and Romain Neugebauer. *simcausal: Simulating Longitudinal Data with Causal Inference Applications*. R package version 0.5.0. 2015. URL: http://CRAN.R-project.org/package=simcausal.

[116] Oleg Sofrygin and Mark J. van der Laan. *tmlenet: Targeted Maximum Likelihood Estimation for Network Data*. R package version 0.1.0. 2015. URL: http://CRAN.R-project.org/package=tmlenet.

[117] Simon Spencer. *interventionalDBN: Interventional Inference for Dynamic Bayesian Networks*. R package version 1.2.2. 2014. URL: https://CRAN.R-project.org/package=interventionalDBN.

[118] Christoph Stadtfeld. *NetSim: A Social Networks Simulation Tool in R*. R package version 0.9. 2013. URL: https://CRAN.R-project.org/package=NetSim.

[119] Christian Steglich, Tom A B Snijders, and Michael Pearson. "Dynamic networks and behavior: Separating selection from influence". In: *Sociological Methodology* 40.1 (2010), pp. 329–393.

[120] Eric J Tchetgen Tchetgen and Tyler J VanderWeele. "On causal inference in the presence of interference". In: *Statistical Methods in Medical Research* 21.1 (2012), pp. 55–75.

[121] Panos Toulis and Edward Kao. "Estimation of causal peer influence effects". In: *Proceedings of The 30th International Conference on Machine Learning* (2013), pp. 1489–1497.

[122] Johan Ugander et al. "Graph cluster randomization: Network exposure to multiple universes". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), pp. 329–337.

[123] A W van der Vaart. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press, 1998. ISBN: 0-521-49603-9.

[124] Michael Væth and Eva Skovlund. "A Simple Approach to Power and Sample Size Calculations in Logistic Regression and Cox Regression Models". In: *Statistics in Medicine* 23.11 (2004), pp. 1781–1792.

[125] Tyler J VanderWeele. "Inference for influence over multiple degrees of separation on a social network". In: *Statistics in Medicine* 32.4 (2013), pp. 591–596.

[126] Tyler J VanderWeele. "Marginal Structural Models for the Estimation of Direct and Indirect Effects". In: *Epidemiology* 20.1 (2009), pp. 18–26.

[127] Tyler J VanderWeele. "Sensitivity analysis for contagion effects in social networks". In: *Sociological Methods & Research* 40.2 (2011), pp. 240–255.

[128] Tyler J VanderWeele and Weihua An. "Social networks and causal inference". In: *Handbook of Causal Analysis for Social Research*. Springer, 2013, pp. 353–374.

[129] Tyler J VanderWeele, Elizabeth L Ogburn, and Eric J Tchetgen Tchetgen. "Why and when" flawed" social network analyses still yield valid tests of no contagion". In: *Statistics, Politics, and Policy* 3.1 (2012).

[130] Tyler J VanderWeele, Eric J Tchetgen Tchetgen, and M Elizabeth Halloran. "Interference and sensitivity analysis". In: *Statistical Science* 29.4 (2014), pp. 687–706.

[131] Tyler J VanderWeele and Stijn Vansteelandt. "Mediation Analysis with Multiple Mediators". In: *Epidemiologic methods* 2.1 (2014), pp. 95–115.

[132] George Vega Yon et al. *netdiffuseR: Network Analysis for Diffusion of Innovations*. R package version 1.16.2. 2016. URL: `https://CRAN.R-project.org/package=netdiffuseR`.

[133] Dylan Walker and Lev Muchnik. "Design of Randomized Experiments in Networks". In: *Proceedings of the IEEE* 102.12 (Dec. 2014), pp. 1940–1951. ISSN: 0018-9219. DOI: `10.1109/JPROC.2014.2363674`.

[134] Pei Wang, Dennis Chao, and Li Hsu. *LogitNet: Infer network based on binary arrays using regularized logistic regression*. R package version 0.1-1. 2009. URL: `https://CRAN.R-project.org/package=LogitNet`.

[135] Duncan J Watts and Steven H Strogatz. "Collective Dynamics of 'Small-World' Networks". In: *Nature* 393.6684 (1998), pp. 440–442.

[136] Zhuo Yu and Mark J van der Laan. "Measuring treatment effects using semiparametric models". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 136 (2003).

[137] Wenjing Zheng and Mark J van der Laan. "Causal mediation in a survival setting with time-dependent mediators". In: *U.C. Berkeley Division of Biostatistics Working Paper Series* Working Paper 295 (2012).