Development of a Multiply Robust Estimator for Sequentially Randomized Trials and Observational Data

A THESIS PRESENTED
BY
Morgan F. Breitmeyer
TO
The Departments of Statistics and Mathematics

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Bachelor of the Arts
IN THE SUBJECT OF
Statistics and Mathematics

Harvard University
Cambridge, Massachusetts
April 2017

Thesis Advisors: Professors James Robins and Miguel Hernan Morgan F. Breitmeyer
Department Advisor: Professor David Harrington

Development of a Multiply Robust Estimator for Sequentially Randomized Trials and Observational Data

Abstract

The statistical field of causal inferences seeks to develop methods that under certain assumptions use observational data to estimate causal effects without the need for costly and time-intensive randomized controlled trials. Two such methods, the g-formula and a doubly robust estimator, are investigated and compared in this thesis, particularly with time-varying treatment regimens. The g-formula is computationally intensive and has the potential to be biased by the incorrect specification of models for outcome and confounders. The doubly robust method is stronger than the g-formula at detecting underlying correlation between outcome and treatment status. While the doubly robust method has slightly higher variance than the g-formula, it is much more robust to errors in model specification. It is also even more robust than initially thought, allowing for significant misspecification of models on the part of the researcher, as shown herein. The doubly robust method is not only multiply robust, but having never before been implemented in the programming language Python, this new program is also significantly more computationally efficient. Furthermore, this first ever implementation in Python is expected to decrease many barriers to use for the scientific community, opening pathways to countless discoveries from massive amounts of data currently available and underutilized.

Contents

This is the dedication.

Acknowledgments

I would firstly like to thank, my advisors, Professors James Robins, Miguel Hernan and David Harrington. Without the support and previous work of Professors Robins and Hernan, this thesis would not have been possible. They guided me through this process and taught me many important things. To Dave, I would like to say thank you for being my advisor and mentor for the last few years. Without you, I would not be studying statistics or have had many of the wonderful opportunities that have been presented to me.

I would also like to thank my family and friends. My mother has worked tirelessly to raise James and me, and for that, we are forever grateful. I would also like to thank her for teaching me about medicine and research my entire life, helping me to understand the framework of my thesis. To my friends, thanks for being endlessly supportive while I wrote my whole thesis, for bringing me snacks and coffee, and for forcing me to leave the library on occasion.

I would finally like to thank all the people who spent time reading and editing bits and pieces of my thesis. I truly appreciate all the help.

Listing of figures

Listing of tables

1

Introduction

Causation versus association – the age-old debate rages on among statisticians, scientists, and students alike. Correlation is easier to understand: are two or more things related to each other? Do taller people weigh more than shorter people? Is the temperature colder when there is snow on the ground than when there is not? Do people who drink red wine and eat dark chocolate have healthier hearts?

Proving correlation is simpler than causation. Select a group of random, unrelated people who drink red wine and eat dark chocolate and a second group of random, unrelated people who do not, and compare their resting heart rates and HDL ("good" cholesterol) levels. If the red wine drinkers and chocolate consumers have better heart health, then it can be concluded that consuming red wine and dark chocolate is associated with a healthier heart. But, there is a catch. This exercise only demonstrates that consuming red wine and chocolate is correlated with a healthy heart; it does not prove that red wine and chocolate actually cause heart health. Perhaps everyone who consumes red wine and dark chocolate also exercises more frequently, is of a healthier weight, or is younger than those who do not, all possible factors that could lead to a healthier heart separate from an individual's red wine and chocolate habits. The association found between dark chocolate and red wine consumption could be reflective of these other healthy heart habits rather than an indication of red wine and dark chocolate driving heart health. This exercise of association does not isolate the underlying cause of the difference between the two groups.

This begs the question of causation; how can one actually prove that the red wine and the dark chocolate cause heart health? Answering this question is not as simple as showing correlation, and, traditionally, a randomized trial is required. A randomized trial experiment involves enrolling a sufficient number of patients, who are randomly assigned into one of two groups: those who are told to drink red wine and eat dark chocolate (the treatment group) and those who are told not to (the controls). A trial like this would likely go on for some extended period of time, maybe years. The individuals in both groups would have their heart rate and their HDL levels measured along the way. Using this data, researchers could quantify the difference in outcome between treatment and control groups to demonstrate an association between treatment and outcome. The randomized nature of this trial is what allows researchers to conclude that the association between consuming red wine and dark chocolate and heart health is actually causation.* However, this process of the controlled randomized trial is complicated, costly, and, until recently, often unfeasible or unethical.

Randomized controlled trials are commonly used in medicine to test the efficacy of a treatment drug. In the US, it takes 12 years and an average of more than $350 million to undergo drug testing and FDA approval.[4] FDA approval requires proof of statistically significant causation of the drug's efficacy through at least two randomized controlled trials. The difficulty surrounding these trials makes the medical innovation process terribly slow, expensive, and inefficient, purely because the current means of proving causation are so challenging.

Further complicating this process, a more complex randomized control trial exists in the form of the sequentially randomized trial. In these trials, several points of randomization exist, leading to more groups than just the control and the treatment groups. Even more, the random mechanism for assigning treatment or control typically varies by group. For instance, consider a trial studying ovarian cancer treatments, where researchers want to test the efficacy of several drugs on different stages of the cancer and want to use a more intense regimen for those with life threatening disease. The researchers decide to only accept subjects whose cancer is in stage 1 or 2, and for those individuals at each time point, the decision to assign treatment or control is entirely random. However, as the trial continues, some subjects' cancer progresses to stage 3 or 4, so the doctors want to increase

*This idea is discussed further in Section 2.3.

the drug dosage. For those subjects, researchers again randomly assign subjects to a different more aggressive treatment or to the same regimen. Because the prognosis is more pressing at this point, the researchers may decide that it is important to switch treatments for those individuals. By the design of the study, this reassignment must again be done randomly, but researchers could indicate that this switch in treatment is a priority, influencing the likelihood that the random assignment results in a treatment change. At each checkup, the researchers perform several tests, such as cancer size, metastatic progression, and morbidity, and using this information, they reassess whether an individual should be switched from the less aggressive treatment group to the more aggressive treatment group. Based off of that additional information, subjects are again assigned a treatment regimen. This progression of reassessment and random reassignment continues throughout the trial. Sequentially randomized trials can be even more sophisticated with numerous treatment types and dispensation strategies.[2]

A sequentially randomized trial such as this one is more effective and intends to provide a more nuanced understanding of a treatment's efficacy. However, this comes at the cost of being quite complicated analytically; conclusions are difficult to define. No longer can researchers perform a simple test on an outcome measure between the control and treatment groups. Instead, with many more and varied treatments over time, the procedures and analysis become even more complicated. While still expensive and time consuming, isolating a causal effect is no longer a simple calculation.

In the last few decades, an alternative solution to these analytical problems has emerged in a field of statistics known as causal inference. It creates and studies various methods to establish causation by means other than the traditional randomized trial. This field emphasizes methods that can work strictly from observational data, or data that contains more sophisticated frameworks. Some examples of observational data include hospital data accumulated over millions of patients and years that track symptoms, level types, outcomes, and demographic surveys, as well as data from other experimental trials. Deploying these alternative methods could decrease the cost and time for proving causation in drug trials, thereby impacting millions of lives.

This thesis is a comparative examination of two distinct methods of causal inference: the g-formula and the doubly robust estimator. Both methods utilize observational data to establish rigorous measures of causation without the traditional mechanisms of randomized trials. They also have the capacity to prove causal effects within dynamic treatment regimens, such as those seen in the sequentially randomized trial, an improvement not seen in former methods of causal inference. The doubly robust estimator is an improved development on the g-formula and is shown to be much more effective at correctly approximating causal effect. It will be shown that this method can withstand significant error caused by human inaccuracy in model selection, resulting in a more consistently precise estimator.

In order to draw causal conclusions using these methods, certain requirements for the context of the data must be fulfilled. If these assumptions are satisfied, these methods can be used on either sequentially randomized trials such as the one for ovarian cancer discussed above or observational data. For instance, there must be a sufficient, albeit small, number of individuals given each possible treatment pathway. Furthermore, group assignment methods, however complex, must be randomized such that no single group is more or less exposed to exogenous survival factors such as age, gender, general health, or morbidity. Moreover, the treatments given must be consistent and well regulated so that the results are clear – for example, the possible pathways of treatment must be predetermined and not altered at the last minute. Finally, the most difficult assumption to meet is that sufficient data exists in order to properly adjust for all common causes of the treatment and the outcome. The methods use this information in order to isolate the causation from other influences. In the study on red wine and chocolate drinking, these are factors such as subject weight, age, exercise habits, smoking, medical history, consumption history of red wine and chocolate, and many other possible factors. This assumption is the most difficult to meet and impossible to guarantee.

In short, with more efficiency, efficacy, and ease, these more advanced methods have widespread applicability in medicine, as novel discoveries arise out of years of underutilized data. However, the validity of their results is contingent on a stringent set of assumptions being met. If these conditions are satisfied, the potential impact is significant from the study of observational data in addition to the implications for the study of more sophisticated trial types. Firstly, the cost savings are of unbelievable magnitude. If causation can be established without years of clinical trials, both governments and research organizations will save millions of dollars. Secondly, the typical timeline of a clinical trial could be drastically decreased using these methods. Observational data already dates back many years and causal results could be acquired in days rather than years, allowing for more efficient medical innovation at a significantly more rapid pace. Thirdly, this method has never before been implemented in Python or in a parallelized fashion, so it can now be applied to very large data sets and in an efficient manner. The popular idea of big data applies here, with ever-growing databases of medical history data and little methodology in place to process it. Previous to this, performing such calculations using other statistical platforms would either take days to weeks or be computationally impossible. This could contribute to managing the ever-growing databases of medical history data, which currently have little methodology in place to process it efficiently. Finally, a key agenda of the National Academy of Sciences and the Humane Society is to find suitable alternatives to animal testing for drug trials in the face of an inefficient medical system in the United States.[16,5] This method provides the framework to replace or supplement many clinical trials, preventing some need for animal testing. All of these innovations could revolutionize the study of modern medicine; nevertheless, parameters for this method, as discussed above, must be rigorously applied in order to effectively establish causal effects. Furthermore, while this thesis applies causal inference methods to medicine and clinical trials, these methods have powerful external applicability. They can be applied to many other fields, including the social sciences, hard sciences, and economics.

2

Background

The randomized control trial introduced in the Introduction seeks to prove causation, allowing for concrete recommendations about the best course of treatment. The randomization in these trials is what allows the conclusion of causation to be drawn, rather than just correlation. Randomization intends to eliminate the potential bias caused by factors other than the treatment of interest. By randomizing subjects into control and treatment groups, neither group should be more likely to respond to treatment. Because of this, any difference that is seen should be purely due to the treatment and not exogenous factors.

Randomized control trials are not only costly and slow, but they can also be unethical. For example, forcing subjects to smoke cigarettes in order to determine if smoking causes lung cancer is not morally appropriate. As such, the motivation to use observational data to answer such causation questions is strong. The field of causal inference seeks to accomplish exactly this for both simple observational data and for data from complex dynamically assigned randomized trial. The field of causal inference began with Neyman in 1990 and Fisher in 1925, who performed agricultural studies to test the efficacy of certain fertilizers on crop yields.[6,7,21] Beginning in the 1970s, Rubin advanced and formalized the field, providing more structured frameworks for understanding and computing causal effects.[15,17,18,20] The methods discussed in this thesis and introduced formally in Chapter 3, the g-formula and doubly robust estimators, were introduced by Robins in 1986 and 2005 respectively.[1,14]

2.1 Causal Effects

In order to translate these ideas of causal inference into statistical terms, some notation must be introduced. The random variable A represents the treatment status, where a value of 1 indicates treated and a value of 0 indicates untreated. A fixed A, which has constant treatment over time, is written as A_i for an individual $i \in \{0, 1, 2, \ldots n\}$ with n the total number of individuals.[*] The random variable

[*] Non-fixed A representations are common and discussed in greater detail in Section 2.1.1.

Y is the outcome variable, often with a value of 0 indicating survival and a value of 1 indicating death. These interpretations of A and Y correspond to the above understanding of causation studies, but for various causal inference studies, the form of Y and A in particular can change depending on the question of interest. For example, Y can be a continuous variable, such as the weight difference of an individual in a weight loss trial or the change in HDL levels in a cholesterol study.

To study the causal effect of A on Y, the desired value is the difference in Y under the varying conditions of A. Notationally, this is the difference between $\mathsf{Y}^{a=1}$[†], the outcome that would be observed under treatment, and $\mathsf{Y}^{a=0}$, the outcome that would be observed under no treatment. This idea of a would be observed value is referred to as the counterfactual, and this is in comparison to the actual observed outcome of Y or Y^{A}.

A causal effect can be seen on an individual level if the difference $\mathsf{Y}_i^{a=1} \neq \mathsf{Y}_i^{a=0}$ is significant. By considering how each individual's responses to varying treatments differ, causation (or lack thereof) can easily be determined using paired differences of the form

$$\mathsf{Y}_i^{a=1} - \mathsf{Y}_i^{a=0} \tag{2.1}$$

These differences would be tested against the null hypothesis of zero difference in outcome for varying treatments.

However, certain difficulties arise using this method. In many studies, it is impossible to have scenarios of both treatment and no treatment for the same individual, particularly if a potential

[†] Note that lowercase letters signify possible values of the random variable, in comparison to uppercase letters which represent actual observed values

outcome is death. Typically, individuals either have $Y_i^{a=1}$ or $Y_i^{a=0}$, but not both, making it impossible to calculate the paired differences. Therefore, a controlled double blinded experiment is often performed, where each individual is randomly assigned treatment or placebo. In these studies, the statistic of interest is the average causal effect in the population,

$$\mathbb{E}[Y^{a=1}] - \mathbb{E}[Y^{a=0}] \tag{2.2}$$

Mathematically, this is equivalent to

$$\mathbb{E}[Y^{a=1} - Y^{a=0}] \tag{2.3}$$

because the average of differences is equal to the difference of averages.[10] Note, that this is not the same as calculating the mean of paired differences as if each individual had received both treatments at different times to calculate individual causal effects. Rather, the difference in the means of the placebo and treatment groups is being calculated to estimate average causal effect across the population.

2.1.1 Time Varying Data

Not all treatment regimens consist of constant treatment over a set period of time. Therefore, a more complicated time-varying treatment can be considered. This would be written for a single individual as $\overline{A}_k = \{A_0, A_1, \cdots, A_k\}$, with time point $k \in \{0, 1, \ldots K\}$, given K as the maximum time value. The overline on $\overline{A}_k$ indicates the history of values up to and including time point k, and the notation $\overline{A}$ represents the full history. As an example of this, a patient with continuous treatment throughout the whole study would have data $\overline{A} = \{A_0 = 1, A_1 = 1, \dots, A_K = 1\} = \{1, 1, \dots 1\}$, which can also be written as $\overline{A} = \bar{1}$. In this scenario, the average causal effect is instead defined as

$$\mathbb{E}\left[Y^{\bar{a}=\bar{1}}\right] - \mathbb{E}\left[Y^{\bar{a}=\bar{0}}\right] \tag{2.4}$$

2.1.2 Deterministic Treatment Regimens

This time-varying framework of the treatment variable can also be considered for the covariate, L. From this, a dynamic treatment strategy can also be created, such that each possible realization $\bar{a}_k$ is dependent on the treatment and covariate history, $\overline{L}_k$ and $\overline{A}_{k-1}$. This can be written as a set of functions $\{g_k(\bar{a}_{k-1}, \bar{l}_k)\}$ where g_k is the function making the treatment decision.

2.1.3 Sequentially Randomized Trial

The sequentially randomized trial is a specific type of treatment regimen, in which a subject's treatment is chosen at each time from an associated density $f(a_k \mid \bar{l}_k, \bar{a}_{k-1})$ for $\bar{a}_k \in \overline{\mathcal{A}}_k$ where $\overline{\mathcal{A}}_k$ is the support of $\overline{A}_k$ in time period k.[25] In this type of randomized trial, each A_k for all subjects is chosen as an independent random draw from this type of distribution density. Sequentially randomized trials guarantee the identifiability assumptions of exchangeability and consistency to be discussed in Section 2.3. This is significantly important because the g-formula and doubly robust estimators are the only current methods which can derive causal effect for data with a sequentially randomized trial, particularly important considering how frequently treatments are given in a sequentially randomized fashion in modern medicine.

2.2 Confounding

In observational data, many measured covariates exist, which we call L; these could include subject height, weight, age, smoking status, medical history, family history, past treatments, and blood levels. Each covariate L is something that could possibly be correlated with the outcome Y. The covariate $\overline{L}$ is a measurable proxy for an unmeasured, unknown, and underlying confounder, U, such as a genetic predisposition to specific diseases. This is to prevent U from creating any bias in measures of causal effect; thus, any measured covariate that can be obtained should likely be used with the goal of covering U as much as possible. It has been shown that if an unmeasured confounder is not sufficiently accounted for, there will be bias in the final estimates.[23]

Theoretically, U should directly impact both L and Y, but not directly impact A because A is only directly dependent on its own history and the history of L.[‡] This indicates that there exists a backdoor path between A and Y through U.[24] The expected way to account for the backdoor path caused by U would be to condition on it, but because it is unknown and therefore unmeasurable, this is not possible. Therefore, methods must be used to create this same effect using only L, which will allow for the study of just the causal effect of A on Y. If the effect of U on Y is fully accounted for, then there should be no bias in the estimate of causal effect. Methods for eliminating the effect of U on Y will be discussed in Sections 2.4 and 2.5.

[‡] See the schematic drawing in Figure 3.1 for a visualization of these connections.

In this scenario, L is referred to as a confounder for the effect of A, reflective of the fact that the underlying bias is due to the unknown U, and L is being used to account for that. It can be shown that in order to estimate the joint effect of all A_k correctly, simultaneously, and without bias, it is sufficient (but not necessary) to block all backdoor paths from U to any A_k for all k.[13]

2.3 Identifiability Assumptions

It is sufficient to show that causal effects are valid and identifiable, meaning they have a single measurement of effect, with the following three assumptions: consistency, positivity, and exchangeability.[3,10] Under these three assumptions, the data closely resembles an ideal randomized trial. Through this randomness, causation can be inferred, rather than simply association. Although the methods are directly testing association, these reasonable and founded assumptions allow the tests to measure causation.[§]

It is important also to consider the assumptions of no measurement error and no model misspecification. Measurement error is a common assumption that is hard to quantify, but could lead to significantly biased results. Model specification is in the hands of the researchers and if done incorrectly, will do an insufficient job at adjusting for confounding, resulting in similar biases as created by measurement errors.

[§] These assumptions are very similar to Rubin's exclusion restrictions, including the Stable Unit Treatment Value Assumption (SUTVA).[19] Rubin defines these exclusion restrictions as those which "rely on external, substantive information to rule out the existence of a causal effect of a particular treatment relative to an alternative."[11] SUTVA is the assumption no individual's treatment impacts the outcome of any other subjects.

2.3.1 Consistency

Consistency is the idea that an individual's potential outcome and their observed outcome are equal[3,10]. Specifically, this is

$$\text{If } A_i = a, \text{ then, } Y_i^a = Y^{A_i} = Y_i \tag{2.5}$$

where Y_i^a is individual i's potential outcome and A_i is the observed treatment.

Consistency can deteriorate under the unintentional presence of varying treatment options. For example, this can be violated if different surgeons perform a procedure, different procedures are performed, or if different drugs or doses are administered to treat the same condition. There are many reasons doctors could use a non standard treatment causing these inconsistencies to appear, such as a family history, complications with a procedure or a drug, or another condition. Protection against this is partially in the understanding and reasonable pruning of the data. Furthermore, this can be addressed with clear and precise questions of interest, and hopefully, detailed data that allows for comprehensive refinement. However, this is a very difficult assumption to ensure without meticulous data cleaning and research about the data acquisition. By including as many relevant covariates, L, the goal is to eliminate as much inconsistency as possible, but this assumption is impossible to ever fully guarantee.

This idea can be expanded to time-varying treatment and covariate variables, as follows

$$\text{If } \bar{A}_k = \bar{a}^g_k, \text{ then, } \bar{Y}_{k+1} = \bar{Y}^g_{k+1} \text{ and } \bar{L}_k = \bar{L}^g_k \tag{2.6}$$

where g is the function that specifies treatment over time.[¶]

2.3.2 Exchangeability

Exchangeability is the idea that individuals in either group of a randomized experiment would have had the same response given the other treatment.[10] There should be no bias to either group to respond favorably or not to treatment or lack thereof; in short, the results should be equivalent if any subject is moved from one group to the other. The counterfactual mean should produce the same results regardless of which individuals actually got treatment and which did not. Statistically, this is $P[Y^a = 1 \mid A = 1] = P[Y^a = 1 \mid A = 0] = P[Y^a = 1]$ for both $a = 0$ and $a = 1$. This means that Y^a is independent of A and that the actual treatment A does not predict the outcome under the counterfactual, Y^a.

Given some indicator of prognosis in the form of L, exchangeability is possible for those with similar prognoses, but it becomes problematic across varying prognoses. For example, two individuals with similar medical history would be expected to have the same counterfactual outcome under exchangeability, but comparing the counterfactual outcome of two subjects with pneumonia, one elderly and one a child, would not induce the same exchangeability. Therefore, it is important to

[¶] Note that this g is the origin of the term g-formula.

condition on such confounders, allowing for conditional exchangeability: $P[Y^a = 1 \mid A = a, L = l] = P[Y^a = 1 \mid A \neq a, L = l]$, i.e. $Y^a \perp\!\!\!\perp A \mid L$.[10] Conditional exchangeability guarantees the ability to measure effects using complete data because the effect measured is just that of treatment and not the underlying confounders, since they should be equivalent in both groups by exchangeability.

The use of a randomized trial theoretically creates exchangeability, which is a powerful conclusion that one would like to be able to recreate in observational causal inference studies. By randomly putting subjects into their groups, there should be no reason that the patients differ between the two groups or will respond to treatment differently. However, exchangeability can be obtained in an observational study if $P[A_k = 1]$ depends only on $\{\overline{A}_{k-1}, \overline{L}_k\}$. Thus, the important assumption of exchangeability in regards to observational data is that $P[A_k \mid \overline{A}_{k-1}, \overline{L}_k]$ is independent of U. A consequence of accounting for U using L in this way is that Y is independent $A_k \mid \overline{L}_k, \overline{A}_{k-1}$. This is statistically referred to as having no unmeasured time-varying confounders.

Although guaranteed for fixed treatments, sequential exchangeability is not guaranteed.[24] Approximate exchangeability can be achieved in practice by including as many covariates as is feasibly reasonable, but this is still risky business as there is no known method for computationally measuring or empirically testing sequential exchangeability. For example, a study could include all the confounders regarding medical history, physicians on the case, and symptoms, but could neglect to include a confounder such as who owns a particular piece of equipment and is getting paid for its use, meaning they would be more motivated to prescribe such treatment. Such a confounder would be unexpected and hard to quantify and could cause problems. By including a large number of other covariates, the hope is that it will be approximately accounted for, but this is one of many examples where it is impossible to determine if it will truly be guaranteed conditional exchangeability. However, the assumption of conditional exchangeability is the same for fixed treatment models and is sufficient for determining causal effect.

2.3.3 Positivity

Positivity is the idea that every possible treatment, covariate condition is represented in the data set. This is the condition that all specified conditional probabilities are well-defined, meaning that for every value of the covariate L, there exist subjects with a specified value of a.[9]

$$\text{If } P[L = l] \neq 0 \quad \forall l, \text{ then } \exists P[A = a \mid L = l] > 0 \tag{2.7}$$

This can also be expressed for time-varying treatments as follows,

$$\text{If } P[\overline{L}_k = \bar{l}_k, \overline{A}_{k-1} = \bar{a}_{k-1}] \neq 0 \quad \forall A_k, \text{ then } \exists P[A_k = a_k \mid \overline{L}_k, \overline{A}_{k-1}] > 0 \tag{2.8}$$

2.4 Inverse Probability (IP) Weighting

Many of the concerns discussed above can be addressed by using the method of Inverse Probability (IP) weighting through simulating a pseudo-population, in which every individual has two data inputs, the expected observed outcomes under treatment and under no treatment. The method to do this is to consider a confounder of the data, L, a value which is known before treatment and often factors into the decision to assign treatment. For example, a confounder in a study on a cholesterol drug could be whether the patient is obese or has high blood pressure. By creating the pseudo-population, the treatment and placebo groups share the same underlying covariate characterizations and distributions.

The pseudo-population can be calculated with the following for each possible combination of A and L,

$$n \cdot P[Y = y \mid A = a, L = l] \cdot P[A = a \mid L = l] \cdot P[L = l] \cdot \frac{1}{P[A = a \mid L = l]} \tag{2.9}$$

where the last term here is the IP weight, $W^A = 1/f(A|L)$.

This weight is equivalent to the inverse of the propensity score, which can be defined as the probability of receiving treatment and written as,[11]

$$e(x) = \frac{N_t(x)}{N_c(x) + N_t(x)} = P[A = a \mid L = l] \tag{2.10}$$

where $x = X_i$ is the true population data and $N_t(x)$ and $N_c(x)$ are the number of individuals in the treatment and control groups, respectively.

The form in expression 2.9 can be used to solve for the standardized mean as follows,

$$\begin{aligned} E[Y^a] &= \sum_l \frac{1}{n} \cdot P[Y = y \mid A = a, L = 1] \cdot P[A = a \mid L = l] \cdot P[L = l] \cdot \frac{1}{P[A = a \mid L = 1]} && (2.11) \\ &= \sum_l \frac{1}{n} \cdot P[Y = y \mid A = a, L = 1] \cdot P[L = l] && (2.12) \\ &= \sum_l E[Y \mid A = a, L = l] P[L = l] && (2.13) \end{aligned}$$

This leads to the confounders being either accounted for or eliminated in the pseudo-population. As a result, the causal effect of A on Y can be effectively estimated using the pseudo-population without any impact from the confounders.

2.4.1 Parametric Estimates

The above non-parametric values for $P[A = a \mid L = l]$ are effective for limited dichotomous confounders, but this method has limitations when L is highly dimensional. To address this, a parametric estimate $\widehat{P}[A = a \mid L = l]$ can be obtained using a logistic regression model for A with all the confounders in L included as covariates.

2.5 Standardization

Like IP weighting, standardization is a method of calculating the marginal counterfactual risk of $P[Y^a = 1]$. This method weights the population by conditioning on the covariates levels in L, in order to make the probability of treatment A independent of the covariates. The weighting can be seen as follows,

$$P[Y^a = 1] = \sum_l P[Y^a = 1 \mid L = l]P[L = l] \tag{2.14}$$

$$= \sum_l P[Y = 1 \mid A = 1, L = l]P[L = l] \tag{2.15}$$

where the equality holds because of the conditional exchangeability. This standardization method can be used to obtain an estimate of the standardized mean,

$$E[Y^a] = \sum_l E[Y \mid L = l, A = a] \cdot P[L = l] \tag{2.16}$$

Note that this returns the same non-parametric expression for the standardized mean as the method of IP weighting because they are mathematically equivalent.

3

Methods

Two formal methods for estimating causal effect are considered in this study: g-formula estimation and doubly robust estimation. These two methods are developments of standardization and IP weighting as discussed in the previous section. They were implemented and studied using simulated data according to the following methods.

3.1 Data Creation

Specifically for the purposes of this study, a data generating algorithm was engineered to provide consistent and easily accessible data for many simulations. The data generated was time-varying and sequentially randomized according to the following schematic. The data serves two purposes: to mimic a sequentially randomized trial and to represent observational data. The data was created as a sequentially randomized trial, but from inspection appears as just observational data with a time-varying treatment, so it can serve as both.

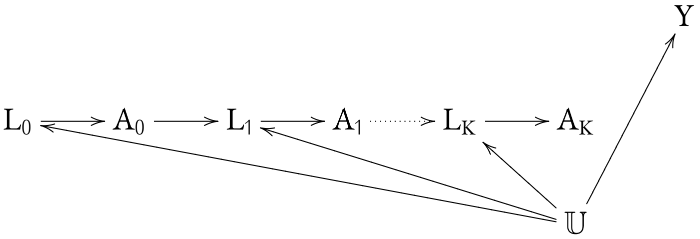


Figure 3.1: Diagram of conditional dependencies in data generating process.

The algorithm to generate datasets is as follows for each individual, of which 1,000 were simulated in this study. The time variable k took on values $\{0, \dots K = 11\}$.

1. Determine the coefficients $\vec{\alpha}$ and $\vec{\beta}$, which are the parameters that define the data generating process in the following models,

$$\text{Logit}[L_{k,i}] = \alpha_0 + \alpha_1 \cdot L_{k-1,i} + \alpha_2 \cdot L_{k-2,i} + \alpha_3 A_{k-1,i} + \alpha_4 A_{k-2,i} + \alpha_5 U_i \tag{3.1}$$

$$\text{Logit}[A_{k,i}] = \beta_0 + \beta_1 L_{k,i} + \beta_2 L_{k-1,i} + \beta_3 A_{k-1,i} + \beta_4 A_{k-2,i} \tag{3.2}$$

These parameters are generated outside the data generating process to provide consistency. In this study, the values were as follows,

$\vec{\alpha}$	$\vec{\beta}$
$\alpha_0 = 0.58986656$	$\beta_0 = 0.17868818$
$\alpha_1 = 0.95344212$	$\beta_1 = 0.89069712$
$\alpha_2 = -0.89822429$	$\beta_2 = 0.89037635$
$\alpha_3 = -0.95566697$	$\beta_3 = 0.20497534$
$\alpha_4 = 0.67520365$	$\beta_4 = 0.10442911$
$\alpha_5 = 2.46365403$	

These coefficients were created by pulling each from $\vec{\alpha}, \vec{\beta} \sim \text{Uniform}(-1.0, 1.0)$ in order to get a variety of positive and negative parameters. The one change made was that α_5 had 1.5 added to the randomly generated value to ensure that the underlying, unmeasured covariate, U, had significant impact for testing purposes.

2. Create the underlying confounder, U_i from $U_i \sim \text{Unif}(0.1, 1)$.

3. Using the logistic expressions 3.1 and 3.2, probabilities for each $L_{k,i}$ and $A_{k,i}$ can be obtained where k is the time and i is the individual, conditional on $\overline{L}_{k-1,i}$ and $\overline{A}_{k-1,i}$. These probabilities are then used to obtain values for $L_{k,i}$ and $A_{k,i}$ using a binomial distribution with the respective probabilities.

   Note that for lower values of time when history is limited, the above expressions are slightly modified as follows,

$$\text{Logit}[L_{0,i}] = \alpha_0 + \alpha_5 U_i \tag{3.3}$$

$$\text{Logit}[A_{0,i}] = \beta_0 + \beta_1 L_{k,i} \tag{3.4}$$

$$\text{Logit}[L_{1,i}] = \alpha_0 + \alpha_1 \cdot L_{k-1,i} + \alpha_3 A_{k-1,i} + \alpha_5 U_i \tag{3.5}$$

$$\text{Logit}[A_{1,i}] = \beta_0 + \beta_1 L_{k,i} + \beta_2 L_{k-1,i} + \beta_3 A_{k-1,i} \tag{3.6}$$

4. Obtain a final Y_i value for each individual where $Y_i \sim \text{Binom}(p = \text{expit}(0.5 + U_i))$.

Note that the unmeasured, underlying covariate, U directly influences each L_k and Y, but does not directly influence A_k. This is an important assumption so that Y can be independent of $A_k \mid$

$\overline{A}_{k-1}$, $\overline{L}_k$ such that $\overline{L}_k$ is adjusting for U. This is an assumption that is difficult to guarantee in true observational data.

Furthermore, the final outcome Y value has no directed arrows from any A to Y and therefore, the treatment has no impact on the outcome under the null hypothesis. This was done to ensure that the causal treatment effect would be zero and bias could be easily measured throughout the study. In reality, it is unlikely that one would be testing truly under the null, so a few tests were performed under an alternative hypothesis of treatment effect in Section 4.2.2. In order to induce such a treatment effect, step 4 was changed to $Y_i \sim \text{Binom}(p = \text{expit}(-1{+}U_i{+}A_K{+}\tilde{E}(\overline{A})))$ where $\tilde{E}(\overline{A}) = \frac{1}{K}\sum_{i=1}^{K} A_i$ is the computational mean of the treatments throughout history. It would be of great interest to be able to directly calculate a true causal effect from this derivation of Y_i. However, because all of the treatment variables, A, are indirectly dependent on U and because Y is dependent on both U and A in this scenario, it is not obvious how to calculate this treatment effect directly. Therefore, these tests under the alternative were included not to measure bias under the alternative but to test for any major problems that could occur under the alternative.

3.2 Parametric G-formula

Similar to IP weighting, parametric estimates can be obtained for standardized estimates. An efficient method for doing this is the generalization of standardization to time-varying treatments and confounders, coined the g-formula method by Robins in 1986.[10,14,24] The method can be used for fixed and time-varying treatments in longitudinal studies, and it seeks to estimate the average causal effect of treatment, which can be estimated as

$$\mathbb{E}[Y^{\bar{a}=\bar{1}}] - \mathbb{E}[Y^{\bar{a}=\bar{0}}] \tag{3.7}$$

where the respective $\bar{a} = \bar{1}$ and $\bar{a} = \bar{0}$ signify constant treatment and no treatment over the entire time period.

The g-formula seeks to calculate each standardized mean using the following,

$$\mathbb{E}[Y^{\bar{a}=\bar{1}}] = \sum_{l_i} \mathbb{E}[Y \mid \bar{L}_t, \bar{A}_t] \cdot \prod_{k=0}^{t} P[L_k = l_k \mid \bar{L}_{k-1}, \bar{A}_{k-1}] \tag{3.8}$$

where $\bar{L}_k = \{L_0 = l_0, L_1 = l_1, \cdots, L_k = l_k\}$ and $\bar{A}_k = \{a_0 = 1, a_1 = 1, \cdots, a_k = 1\}$ are the history of the treatment and covariate variables up to and including time k. The equivalent formula can be derived for $\mathbb{E}[Y^{\bar{a}=\bar{0}}]$.

One of the key reasons for using the g-formula method is that it is able to account for time-varying confounders in the presence of treatment-confounder feedback. This is equivalent to each L_k being dependent on A_{k-1}.[14] In these scenarios, traditional methods for adjusting for the confounder, such as stratification, regression, and matching may introduce bias. However, the g-formula method (as well as IP weighting) is able estimate the joint effect of all treatment values $\{A_0, A_1 \ldots A_K\}$ simultaneously and without bias, which these other methods are unable to do.[8,24]

The g-formula method has been shown to have a smaller variance than IP weighting methods, but this comes with added parametric modeling assumptions.[25] The smaller variance is due to the fact that the g-formula uses maximum likelihood estimates, in comparison to the semi-parametric estimator used in IP weighting. Furthermore, IP weighting does fault and become quite unstable under violations (or close violations) of the positivity assumption, due to division by a potentially near zero probability $P[A_k = a_k \mid \overline{L}_k, \overline{A}_{k-1}]$.

These improvements are, however, under the assumption of exchangeability, and the fact that the g-formula relies more heavily on parametric assumptions, which can lead to bias. The presence of bias is dependent on the accuracy of the models for Y and L_k for all k. IP weighting methods are also dependent on the accuracy of their models, just different models, i.e. for A_k conditional on $\overline{L}_k, \overline{A}_{k-1}$.

3.2.1 Protocol

The analysis method is performed in several steps, as follows

1. Create outcome models: Fit models for the outcome variable Y and the covariates, L_k at each time using the original dataset. The model for Y is a regression model on the treatment variable A and the confounders, L.

   In this case, the following models were chosen for $Y \mid \overline{A}_t, \overline{L}_t$ and $L_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}$,

$$\mathrm{Logit}[Y \mid \overline{A}_t, \overline{L}_t] = \theta_0 + \theta_1 A_t + \cdots + \theta_j A_0 + \theta_{j+1} L_t + \cdots + \theta_{j+k} L_0 \tag{3.9}$$

$$\mathrm{Logit}[L_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}] = \gamma_0 + \gamma_1 L_{k-1} + \gamma_2 L_{k-2} + \gamma_3 L_{k-3} + \gamma_4 A_{k-1} + \gamma_5 A_{k-2} + \gamma_6 A_{k-3} \tag{3.10}$$

   A time lag of only three historical values was deemed sufficient for the model of L_k through testing as discussed in Section 4.1.

   Note that for initial time points where there was insufficient history for the full model, smaller models were created as follows

$$\mathrm{Logit}[L_1 \mid L_0, A_0] = \gamma_0' + \gamma_1' L_0 + \gamma_2' A_0 \tag{3.11}$$

$$\mathrm{Logit}[L_2 \mid L_0, L_1, A_0, A_1] = \gamma_0'' + \gamma_1'' L_{k-1} + \gamma_2'' L_{k-2} + \gamma_3'' A_{k-1} + \gamma_4'' A_{k-2} \tag{3.12}$$

2. Predict using Monte Carlo: Using the fitted model created in step 1, predict the counterfactual outcome Y. Using expressions 3.9 through 3.12, a Monte Carlo simulation must be performed. This is because it is impractical to calculate expression 3.10 directly for a continuous L. This process is done as follows for time $k = \{0, \dots, K\}$, and individuals $i = \{1, \dots, n\}$ keeping the test treatment regimen of interest $\bar{a}$ in mind through the process.

   (a) Select the L_0 value from a random individual from $i \in \{1, \dots, n\}$.

   (b) Obtain a probability of L_1 using this L_0 and a_0 in expression 3.11 and then obtain a sample value of L_1 by pulling from a binomial distribution.

   (c) Obtain a probability of L_2 using the L_0, L_1, a_0, and a_1 in expression 3.12 and then obtain a sample value of L_2 by pulling from a binomial distribution.

   (d) Continue the same process until time K using expression 3.10 to get a full history $\overline{L}_K$ and all the probabilities $P[L_k = l_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}]$.

   (e) Using expression 3.9, $\bar{a}$ and the above solved for $\overline{L}_K$, calculate $P[Y \mid \overline{A}_K, \overline{L}_K]$.

   (f) Take the product of all the probabilities $P[L_k = l_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}]$ for $k = 0, \dots, K$ and $\mathbb{P}[Y = 1 \mid \overline{A}_K, \overline{L}_K]$ to get a final estimate.

   (g) Repeat steps (2a) through (2f) for as many simulations as desired. In this study, 10,000 individuals were simulated.

   (h) Take the mean of all simulation values to obtain $\mathbb{E}[Y^{\bar{a}}]$.

   (i) Repeat all above steps for the opposing treatment regimen of interest $\bar{a}'$ and take the difference $\mathbb{E}[Y^{\bar{a}}] - \mathbb{E}[Y^{\bar{a}'}]$ to get the average causal treatment effect.

The Monte Carlo simulation is used to create two new simulated datasets, the first having all individuals under no treatment ($A = 0$) at all times and the second having all individuals treated ($A = 1$) at all times. Each of these new datasets has the same size as the original and the same "individuals", meaning the same covariate L distribution at baseline. The standardized means could be obtained by creating a weighted average for $E[Y^{a=0}]$ from the first new dataset and one for $E[Y^{a=1}]$ from the second new dataset.

3.3 Doubly Robust Estimation

The method of doubly robust estimation, as proposed by Bang and Robins,[1] combines the two previously discussed methods of IP weighting and standardization. IP weighting estimates $P[A = a \mid L = l]$, while standardization estimates $P[Y = 1 \mid A = a, L = l]$ and $P[\overline{L}_k = l_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}]$. Therefore, these two techniques are expected to provide different answers, unless there are no models used to create estimates as would be the case if all estimates were non-parametric.[10]

The method of doubly robust estimation does not make use of the observed data treatment density, as the methods of IP weighting and standardization do. This allows for some use of missing data points, rather than just having to drop these data points. This also prevents a lack of skewing due to overrepresented populations that could result from missing data. It will be shown in Chapter 4 that the estimators derived are consistent if either the model for treatment given the past (as in IP weighting) is correctly specified or the models for the outcome and covariates given the past (as needed to implement the parametric g-formula) are correctly specified, without knowing which is correct. It is only when both models are misspecified that the method breaks. This is the derivation of the term doubly robust. Furthermore, however, evidence will be presented showing that this method is actually more than doubly robust. Previously, it has been shown that either model being correctly specified for all time will prevent the introduction of bias. However, this thesis demonstrates that the model does not have to be correctly specified for the entire time. Section 4.3 presents the results that the two models can be correctly specified in specific order combinations without introducing bias.

3.3.1 Protocol

The method can be performed recursively using the following steps,

1. Build a model for the treatment A_k with data pooled for all time $m \in \{1, \ldots, K\}$ and all individuals $i \in \{1, \ldots n\}$ and obtain the MLE $\hat{\alpha}$ of α using logistic regression.

$$\text{logit}\{P[A_{m,i} = 1 \mid \bar{l}_{m,i}, \bar{a}_{m-1,i}; \alpha]\} = w_m[\bar{l}_{m,i}, \bar{a}_{m-1,i}; \alpha] \qquad (3.13)$$

   This model was as follows,

$$P[A_m = 1 \mid \bar{L}_m, \bar{A}_{m-1}; \hat{\alpha}] = \alpha_0 + \alpha_1 \cdot L_m + \alpha_2 \cdot A_{m-1} + \alpha_3 \cdot L_{m-1} + \alpha_4 \cdot L_{m-2} + \alpha_5 \cdot A_{m-2} \qquad (3.14)$$

2. Set $\hat{T}_{K+1} = Y$.

3. Recurse for $m = K + 1, \ldots, 2$

   (a) Use iteratively re-weighted least squares (IRLS) and a specified parametric regression model for T_m to estimate the regression function

$$h_{m-1}(\bar{L}_{m-1}, \bar{A}_{m-1}; \beta_{m-1}, \phi_{m-1}) = \Psi\{s_{m-1}(\bar{L}_{m-1}, \bar{A}_{m-1}; \beta_{m-1}) + \phi_{m-1}\bar{\pi}_{m-1}^{-1}(\hat{\alpha})\} \qquad (3.15)$$

   which gives the conditional expectation of

$$\mathbb{E}\left[\hat{T}_m \mid \bar{L}_{m-1}, \bar{A}_{m-1}\right] \qquad (3.16)$$

   The known function s_m is specified on a case by case basis, and in this case was chosen to be as follows,

$$s_m(\bar{L}_m, \bar{A}_m; \beta_m) = \beta_0 + \beta_1 L_m + \beta_2 A_m + \beta_3 L_{m-1} + \beta_4 A_{m-1} + \beta_5 L_{m-2} + \beta_6 A_{m-2} \qquad (3.17)$$

   Furthermore, the function $\bar{\pi}_m(\hat{\alpha})$ is the propensity score model and is specified as follows

$$\overline{\pi}_m(\hat{\alpha}) = \prod_{j=1}^{m} f(A_m \mid \overline{L}_m, \overline{A}_{m-1}; \hat{\alpha}) \tag{3.18}$$

$$= \prod_{j=1}^{m} \zeta_0 + \zeta_1 L_m + \zeta_2 L_{m-1} + \zeta_3 A_{m-1} + \zeta_4 A_{m-2} \tag{3.19}$$

The given Ψ is the canonical link function of the chosen GLM.

(b) Let $\hat{h}_{m-1}(\overline{L}_{m-1}, \overline{A}_{m-1}; \hat{\beta}_{m-1}, \hat{\phi}_{m-1})$ be the predicted model derived in step 3a. This implies that $(\hat{\beta}'_{m-1}, \hat{\phi}'_{m-1})$ is a solution of

$$0 = \tilde{\mathbb{E}}\left[\left[\hat{T}_m - \Psi\{s_{m-1}(\overline{L}_{m-1}, \overline{A}_{m-1}; \hat{\beta}_{m-1}) + \hat{\phi}_{m-1}\overline{\pi}^{-1}_{m-1}(\hat{\alpha})\}\right]\left(\frac{\partial s(\overline{L}_{m-1}; \beta_{m-1})}{\partial \beta'_{m-1}, \overline{\pi}^{-1}_{m-1}(\hat{\alpha})}\right)\right] \tag{3.20}$$

where $\tilde{\mathbb{E}}(X) = \frac{1}{n}\sum_{i=1}^{n} X_i$ is the computational average.

(c) Set

$$\hat{T}^{a_{m-1},\ldots,a_K}_{m-1} = \hat{h}_{m-1}(\overline{L}_{m-1}, \overline{A}_{m-2}, a_{m-1}) \tag{3.21}$$

$$= \Psi\{s_{m-1}(\overline{L}_{m-1}, \overline{A}_{m-2}, a_{m-1}; \beta_{m-1}) + \phi_{m-1}\overline{\pi}^{-1}_{m-2}(\hat{\alpha}) f(a_{m-1} \mid \overline{L}_{m-1}, \overline{A}_{m-2}; \hat{\alpha})\} \tag{3.22}$$

where a_{m-1} is our treatment value of interest, the lowercase letter indicating a test value rather than an observed.

4. To estimate the final $\mathbb{E}[Y^{\bar{a}}]$, solve

$$\mathbb{E}[Y^{\bar{a}}] = \tilde{\mathbb{E}}(\hat{T}_1) = \tilde{\mathbb{E}}(\hat{T}^{\bar{a}}_1) \tag{3.23}$$

5. Repeat all above steps for the opposing treatment regimen of interest $\bar{a}'$ and estimate the difference $\mathbb{E}[Y^{\bar{a}}] - \mathbb{E}[Y^{\bar{a}'}]$ to get the average causal treatment effect.

Regarding the Ψ function used in expression 3.15, the desired method to do this is using a GLM with an underlying distribution (or family) of a Gaussian normal and a logit link. However, Python does not have the capacity to do it this way, so alternatives had to be tested and considered, including logistic regression, basic linear regression with an expit applied after step 3c as well as using a logistic regression and taking the predicted probability to pull 1000 samples from a binomial distribution for each individual and regressing off that new data in the next step. However, through much testing, it was concluded that the best means to do this was using a GLM with an underlying binomial distribution and a logit link.

3.4 Variance Estimate

In order to compute the variance of the estimates obtained using the above two methods, non-parametric bootstrapping was used. This was done by repeating the above processes 1,000 times and collecting all of the resulting estimates. The variance of these estimates was then obtained.

3.4.1 Protocol

1. Determine the number of simulations to be performed. In this case, 1,000 simulations were performed.
2. Perform the following steps as many times as decided in step 1
    - (a) Create a dataset using the data generating algorithm described in Section 3.1.
    - (b) Estimate the average causal treatment effect using the g-formula.
    - (c) Estimate the average causal treatment effect using the doubly robust method.
3. Calculate the mean of estimates for each of the two methods
4. Calculate the variance and standard error of each mean of estimates.

Note that it is also possible to directly compute the variance of a doubly robust estimator, but this was beyond the scope of this project. The ability to calculate variance without bootstrapping is highly efficient and proves another advantage of the doubly robust estimator.

4

Results Discussion

The methods described in Chapter 3 were implemented and tested for their efficacy at evaluating average causal treatment effect. Results and discussion are presented below.

4.1 Natural Course

In order to test the specified models used for the g-formula method, a natural course study was performed. This involved simulating the data directly using the models specified for the g-formula method. Additionally, a model for the treatment variable had to be created as well, giving the following three models,[*]

$$\mathrm{Logit}[Y \mid \bar{A}_t, \bar{L}_t] = \theta_0 + \theta_1 A_t + \cdots + \theta_j A_0 + \theta_{j+1} L_t + \cdots + \theta_{j+k} L_0 \tag{4.1}$$

$$\mathrm{logit}[L_k \mid \bar{L}_{k-1}, \bar{A}_{k-1}] = \gamma_0 + \gamma_1 L_{k-1} + \gamma_2 L_{k-2} + \gamma_3 L_{k-3} + \gamma_4 A_{k-1} + \gamma_5 A_{k-2} + \gamma_6 A_{k-3} \tag{4.2}$$

$$\mathrm{logit}[A_k \mid \bar{L}_k, \bar{A}_{k-1}] = \delta_0 + \delta_1 L_k + \delta_2 L_{k-1} + \delta_3 L_{k-2} + \delta_4 A_{k-1} + \delta_5 A_{k-2} + \delta_6 A_{k-2} \tag{4.3}$$

Ten thousand new individuals were simulated to determine the natural course of these models. Several other models were tested as well, such as including more historical terms, less historical terms, interaction terms, squared terms, and cubed terms; however, these models above were the best, showing the least difference between the natural course and the true data. The majority of the other models tried had highly significant p-values of difference between the natural course and the original data.

Table 4.1 below compares the true mean of the original data frame and the average from the data simulated under the natural course under both the null and alternative hypotheses. The models for Y and L in expressions 4.1 and 4.2 respectively are particularly good, showing no significant differ-

[*] Abbreviated models, like those specified in expressions 3.11 and 3.12, were used for time points $t = 0, 1, 2$ since adequate historical data is not available.

ence between the true data and the natural course under both the null and alternative hypotheses, indicating that these are a good choice to use in the g-formula method. The model for A was very difficult to pin down, with every model tried showing highly significant differences to the true data, provoking p-values mostly less than e^{-8}. This model in 4.3 was the only model that had a p-value remotely close to a cutoff of $\alpha = 0.05$, but is still considered significantly different, a point of concern. However, this model was only created in order to simulate the natural and is not actually used in the g-formula since the treatments are set by the researcher. Therefore, although this appears concerning, it is irrelevant to the results and the important conclusion of the natural course is the strength of the models for Y and L.

		Natural Course		Natural Course
Variable	True Mean	Average Mean	p-value	95% Conf. Int.
		Under the Null Hypothesis		
$\mathbb{E}[Y]$	0.736	0.742	0.687	(0.733, 0.750)
$\mathbb{E}[A]$	0.845	0.852	0.027	(0.851, 0.853)
$\mathbb{E}[L]$	0.826	0.828	0.573	(0.826, 0.830)
		Under the Alternative Hypothesis		
$\mathbb{E}[Y]$	0.753	0.765	0.417	(0.756, 0.773)
$\mathbb{E}[A]$	0.842	0.851	0.012	(0.850, 0.852)
$\mathbb{E}[L]$	0.825	0.826	0.721	(0.824, 0.828)

Table 4.1: A table outlining the results of the natural course test of the methods. True mean is specified using the underlying dataset which was the models for the used to simulate the natural course.

4.2 Simulation of the Two Methods

As discussed in Section 3.4, a simulation of 1,000 iterations was performed. Each iteration of this simulation consisted of creating a new dataset and obtaining both the g-formula estimate and the doubly robust estimate for the causal treatment effect.

4.2.1 Under the Null Hypothesis of No Treatment Effect

This was first tested under the null hypothesis of no treatment effect, as detailed by the data generating algorithm in Section 3.1. The results of these simulations can be seen below in Table 4.2 and Figure B.1. These both indicate that the doubly robust method has a more precise average causal treatment effect across simulations.[†] However, this improvement comes with higher variance and bias, as shown in the wider confidence interval and the larger spread on the histogram.

Furthermore, the relationships between the data and the causal effect estimates were examined in Figure 4.1. Of note, the doubly robust method is able to capture correlation between Y and A which the g-formula method does not, as evidenced by the first row of plots. This is an important indicator that the doubly robust method more successfully picks up treatment effect. This likely contributes to the more precise causal treatment effect measure. Furthermore, the higher variance in the estimate may

[†] As discussed in Section 3.1, no effect of A on Y was included under the null hypothesis, so the true causal treatment effect should be zero.

Method	Average Causal Treatment Effect	Average Bias	Variance of Effect Estimate	95% Conf. Int. of Effect Estimate
		Under the Null Hypothesis		
G-Formula	0.0033	0.016	0.00024	(-0.00063,0.00128)
Doubly Robust	0.00026	0.031	0.00079	(-0.00149, 0.00212)
		Under the Alternative Hypothesis		
G-Formula	0.0005	NA	$6.5e^{-8}$	(0.00047,0.00050)
Doubly Robust	0.222	NA	0.0009	(0.2204, 0.2243)

Table 4.2: A table presenting the results of 1,000 data simulations under the null and alternative hypotheses, within which the g-formula and doubly robust estimators of average causal effect were each obtained. Average causal treatment effect is the mean of the causal treatment effect across the simulations. The variance and confidence intervals reflect the variability in that causal treatment effect over 1,000 simulations. There is no bias measurement under the alternative hypothesis because of the noted inability to have a true measure of causal effect.

4.2.2 Under the Alternative Hypothesis of Treatment

The above simulation was also tested under an alternative hypothesis of some treatment effect. The effect was induced in the data creation algorithm by changing step 4 to instead be $Y_i \sim \text{Binom}(p = \text{expit}(-1 + U_i + A_K + \tilde{E}(\overline{A})))$ where $\tilde{E}(\overline{A}) = \frac{1}{K}\sum_{i=1}^{K} A_i$ is the computational mean of the treatments throughout history. This was in order to guarantee that Y is dependent on $\overline{A}$, inducing a significant treatment effect.

This second set of simulation testing was performed in order to determine if the methods become biased under the alternative hypothesis. As noted in Section 3.1, it is not possible to directly calculate a true causal effect of A on Y because of the indirect dependence between A and U. This is at the core of why alternative methods for calculating such effects are necessary. Because it is impossible to get an unbiased measure of the treatment effect, no biases can be calculated. However, the

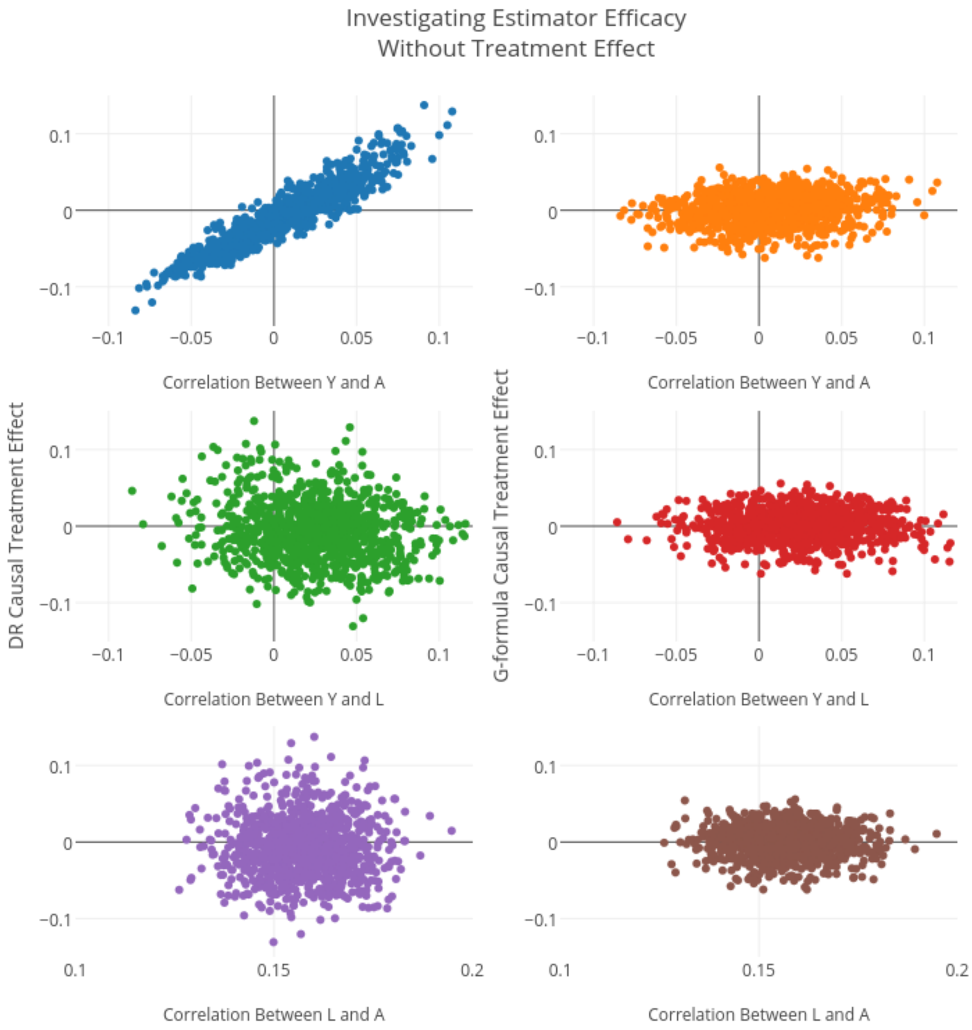


Figure 4.1: Various plots showing the relationship between underlying data correlations and estimated causal treatment effects using the two methods. Each data point is representative of one dataset. The same 1,000 datasets are used for each effect estimation.

results do present a few interesting findings. Firstly, the results in Table 4.2 and Figure 4.2 show that there appears to be significant issues with the g-formula under the alternative hypothesis, notably that the estimates are all zero with extreme precision and no variance. The g-formula seems to fail to detect any treatment effect, despite the data having a definite treatment effect. The top right graph in Figures 4.1 and 4.2 show that the estimates seem to be detecting none of the correlation between Y and A. Considering that the method calculated quite similar, and seemingly more accurate, estimates under the alternative hypothesis as under the null, these results indicate a breakage in the g-formula method. This is in line with the fact that under the null, the g-formula was capturing almost none of the correlation between Y and A. The question of what is causing this could be a mistake in the coding, a fault in the design of the models, or a violation of assumptions.

On the other hand, the doubly robust estimator appears to display a reasonable estimate for the causal effect. It is unclear if there is any bias in this estimate since there is no calculable true causal effect estimate. Figure 4.2 demonstrates that under the alternative, this method is again strongly capturing correlation between Y and A. As under the null hypothesis, the estimates seem to show low variance, indicating a low impact of randomness in the outcome. This is another benefit of the doubly robust estimator over the g-formula, in that it does not require the incorporation of a Monte Carlo simulation which can increase variance due to randomness.

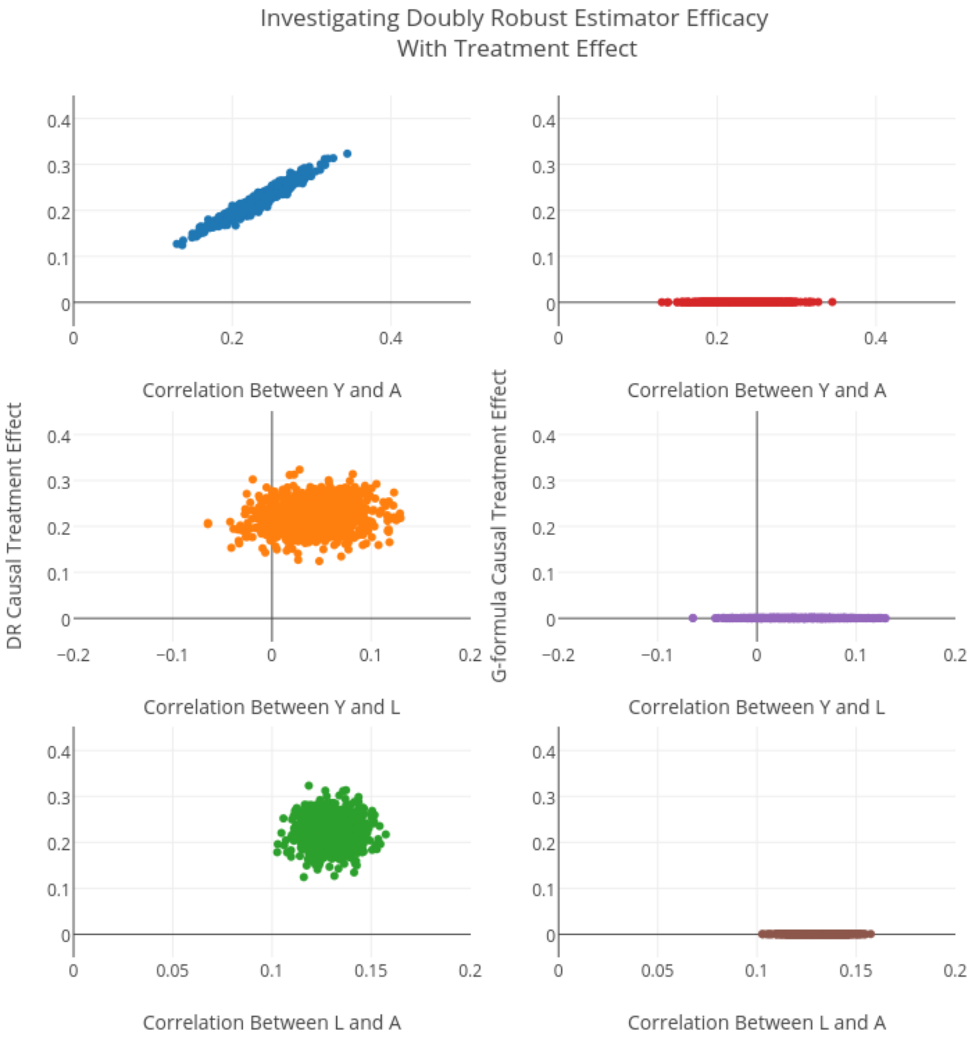


Figure 4.2: Various plots showing the relationship between underlying data correlations and estimated causal treatment effects using the two methods. Each data point is representative of one dataset. The same 1,000 datasets are used for each effect estimation.

4.2.3 Run Times

In order to test the true efficiency of running these methods, run times were calculated across the 1000 simulations under both the null and alternative hypotheses. It has traditionally been the belief that the g-formula is less efficient than the doubly robust estimator because of the Monte Carlo simulation. However, this new implementation of it in Python was through parallelization, and as a result, it is significantly faster. Table 4.3 shows that it takes an average of 0.38 seconds to get a g-formula estimator compared to 2.03 seconds for the doubly robust estimator.

Method	Total Run Time	Average Run Time
Data Generating Algorithm	2 min, 15 sec	0.14 sec
	Under the Null Hypothesis	
G-Formula	6 min, 19 sec	0.38 sec
Doubly Robust	33 min, 50 sec	2.03 sec
	Under the Alternative Hypothesis	
G-Formula	6 min, 27 sec	0.39 sec
Doubly Robust	34 min, 5 sec	2.05 sec

Table 4.3: This table shows the run times of the two methods implemented in Python, as well as the data generating algorithm.

4.3 Confirming Double Robustness

Twenty-five simulations were performed in order to test the double robustness of the “doubly” robust estimator. Four different setups were tested:

- Using the correctly specified model for both user specified functions, $\hat{\alpha}$ and s_{m-1}
- Using an intentionally misspecified model for $\hat{\alpha}$ while keeping the correct model for s_{m-1}
- Using an intentionally misspecified model for s_{m-1} while keeping the correct model for $\hat{\alpha}$
- Using intentionally misspecified models for both $\hat{\alpha}$ and s_{m-1}

The misspecified models used were as follows,

$$f(A_m \mid \bar{L}_m, \bar{A}_{m-1}; \hat{\alpha}) = \alpha'_0 + \alpha'_1 \cdot L_{m-3} + \alpha'_2 \cdot A_{m-3} \tag{4.4}$$

$$s_m(\bar{L}_m, \bar{A}_m; \beta_m) = \beta'_0 + \beta'_1 A_m + \beta'_2 L_m + \beta'_3 A_{m-4} + \beta'_4 L_{m-4} \tag{4.5}$$

These models were intentionally chosen because they are unlikely to be strong models for prediction based on the knowledge of how the data was created. In practice, one would hope that the researcher using these methods would be wiser as not to use such terrible models, particularly for both models. This testing is to demonstrate how robust the method is to misspecification, or how accessible the method is to those with less versed statistical knowledge as they may be more inclined to make minor mis-specifications in the model.

The results of this testing are shown in Table 4.4 and this shows evidence that the method is indeed doubly robust. This means that when one model is incorrect, but the other is correct, then the estimate should not be biased. The average causal treatment effect across the simulations is not significantly impacted when either the $\hat{\alpha}$ model or the s_{m-1} model is incorrectly specified. When both models are misspecified, statistically significant bias is introduced (p-value 0.005). Furthermore, both Table 4.4 and Figure 4.3 show that the variance is significantly higher when both models are misspecified. From the boxplot, it can be seen that the variance stays quite consistent when only one model is misspecified but is much larger when both are misspecified.

Models	Average Bias in Causal Treatment Effect	p-value	Standard Error of Bias
Both models correctly specified	0.027	NA	0.0039
$\hat{\alpha}$ correct, s_{m-1} misspecified	0.030	0.579	0.0041
s_{m-1} correct, $\hat{\alpha}$ misspecified	0.030	0.521	0.0042
Both models misspecified	0.133	0.005	0.0355

Table 4.4: Table showing the results of testing the double robustness of the model. The p-value is from a two-sample t-test comparing the bias from misspecified versions with the bias results from having both models correctly specified.

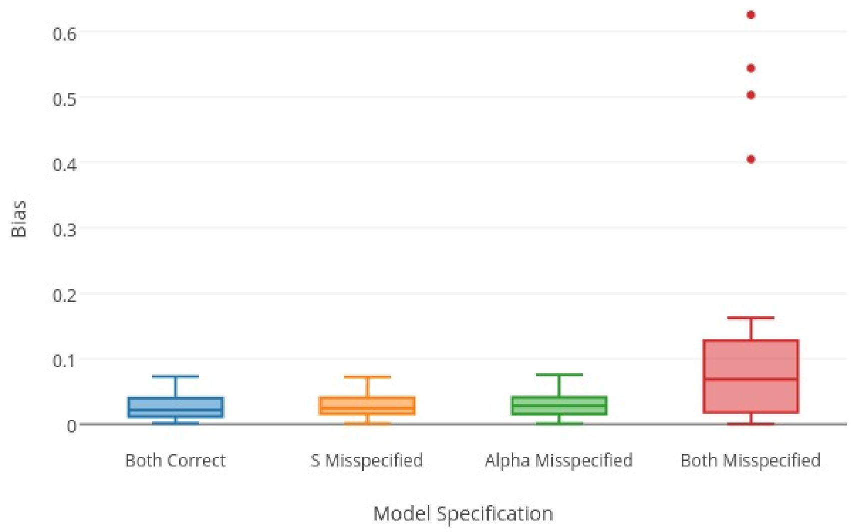


Figure 4.3: This boxplot shows the bias in the causal treatment effect estimates for varying combinations of models correctly specified. The fourth box, when both models are misspecified, shows the strongest bias and the most variance in that bias. However, the model remains robust when only one model is misspecified.

4.4 Testing Multiple Robustness

Having established that the model was indeed doubly robust, the model was then tested to see if it was multiply robust. Several different combinations of correctly specified models were tested, namely of the form where the below specified models were correct and the models not listed were incorrect.

$$\pi_3, \dots, \pi_{j-1}, s_j, \dots, s_K \text{ for } j \in \{3, \dots, 12\} \tag{4.6}$$

$$s_3, \dots, s_{j-1}, \pi_j, \dots, \pi_K \text{ for } j \in \{3, \dots, 12\} \tag{4.7}$$

$$s_3, \pi_4, s_5, \pi_6, s_7, \pi_8, s_9, \pi_{10}, s_{11} \tag{4.8}$$

$$\pi_3, s_4, \pi_5, s_6, \pi_7, s_8, \pi_9, s_{10}, \pi_{11} \tag{4.9}$$

Twenty-five simulations were performed, for which the causal treatment effect was estimated for each of these varying combinations of correctly specified models using the doubly robust method. Then, the bias was obtained across the twenty-five estimates. The results in Figure 4.4 show that when the correctly specified models are of the form $s_3, \dots, s_{j-1}, \pi_j, \dots, \pi_K$ for $\forall j \in \{4, \dots, 12\}$, significant bias is introduced.[‡] However, no bias is introduced when the correctly specified models are of the form $\pi_3, \dots, \pi_{j-1}, s_j, \dots, s_K$ for $\forall j \in \{4, \dots, 12\}$. This implies that the doubly robust method is actually more robust than just doubly. A Tukey test was also performed to test this, and the results are shown in Figure 4.5. The statistically significantly biased model combinations confirm the above observations.

[‡] More specific values for the bias across all these models can be seen in Table B.1 in Appendix B.

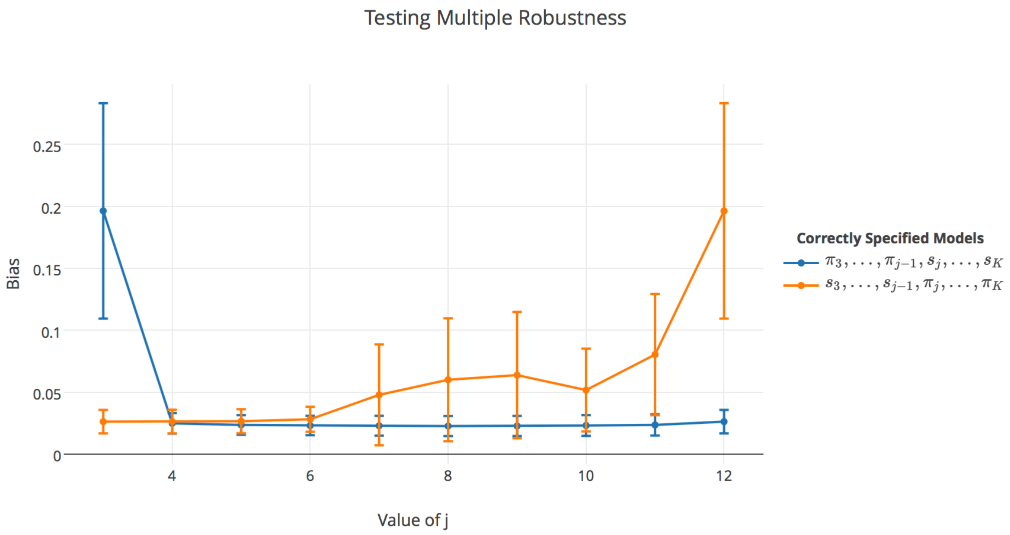


Figure 4.4: Plot showing the results of testing for multi-robustness. The models are correctly specified as given in the legend. Note that the end points of $j = 3$ and $j = 12$ correspond to the opposite end for the other line and represent the model where only one of the two models is correctly specified throughout.

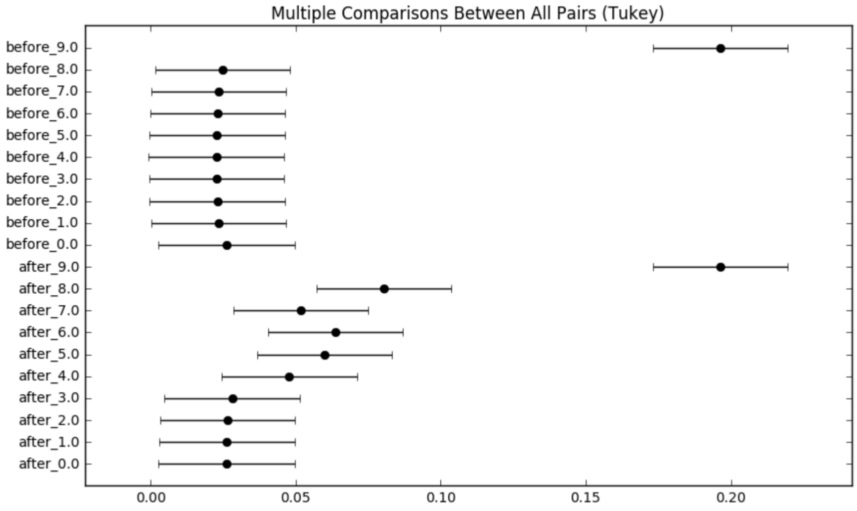


Figure 4.5: A plot of the confidence intervals of each of the model specifications in the testing of multiple robustness. The y-axis indicates which model, where the number is the value of j in the following models, the before model is $\pi_3, \dots, \pi_{j-1}, s_j, ..., s_K$ and the after model is $s_3, \dots, s_{j-1}, \pi_j, \dots, \pi_K$. The before and after notation signifies whether the π models are correctly specified in order before or after the s_m models. The before values correspond to the blue lines in Figure 4.4 while the after values correspond to the orange line.

5

Conclusion

In conclusion, two methods of estimating causal treatment effect are presented here: the g-formula and the doubly robust estimators. The evidence presented leads to the conclusion that the method of doubly robust estimation could aptly be renamed multiply robust estimation, as it can robustly withstand substantial variation and misspecification in model selection. The doubly robust estimator is perhaps not as precise as the g-formula under the null hypothesis, but it is more robust to user model misspecification. Furthermore, although the bias is unmeasurable under the alternative hypothesis, the doubly robust shows no pressing issues such as those seen with the g-formula under the alternative hypothesis.

5.1 Contributions

These methods have never been implemented in this form in Python before, an advance that will significantly increase the efficiency and accessibility of use, decreasing barriers between scientific study and causal inference. The functions currently created and listed in Appendix A could be easily restructured into a software package in order to be widely used and implemented. Capability must also be extended for including an extensive number of covariates, an important assumption of the model.

The Python implementations make both methods highly efficient, but the difference seen is more significant for the g-formula. The doubly robust estimator has previously been considered a more efficient means of estimating treatment effects because of the recursive nature, in comparison to the g-formula which has suffered inefficiencies due to the high number of replicates required for the Monte Carlo simulation. Of note, the parallelization of the Monte Carlo simulation in this implementation actually improves efficiency significantly, making the efficiency difference between the two methods less concerning.

This thesis first confirmed that the doubly robust estimator is indeed doubly robust, meaning that as long as at least one of the two models contained within is correctly specified, the treatment effect remains unbiased. Furthermore, it was demonstrated that the doubly robust estimator is actually more robust than thought, remaining unbiased as long as the π models are correctly specified before the s models chronologically.

In total, the two methods were implemented in Python, and their results compared across several criteria. The g-formula is quicker, but prone to error from user misspecification and problems under the alternative hypothesis of a treatment effect. On the other hand, the doubly robust estimator is more accurate albeit with slightly higher variance across simulations, but it is significantly more robust to situations that could have the potential to introduce bias, such as slight user misspecification or in the scenario of a treatment effect.

5.2 Limitations

Both methods studied require a stringent set of assumptions, many of which are difficult to confirm. Because of this, finding the right situation in which to use these methods can be difficult. Many of the assumptions can be approximately achieved, although the inability to confirm their fulfillment makes it difficult to calculate the bias of these approximations.

5.3 Implications

The evidence presented here regarding the increased efficiency of the two methods' implementation in Python is likely to have an impact on many fields of science and medicine. This implementation could be efficiently applied to massive datasets, often used in medical research in pursuit of FDA approval, possibly proving many novel discoveries. These methods can provide significant utility for the astonishing amounts of data that is not frequently capitalized upon. Even if the assumptions are difficult to confirm, it would serve as a point of positive progress towards effective use of all of this data. This application alone could potentially save the fields of medicine and healthcare millions, even billions, of dollars in years to come.

Furthermore, the robustness of the doubly robust method presents it as an improved estimator for research purposes because it can sustain substantial errors caused by the researcher. The method can withstand significant model misspecification without introducing bias, meaning that researchers do not need perfect statistical knowledge in order to use this method. This makes it more accessible to users of all backgrounds.

Finally, in addition to the implications for observational data, this method is an important development in the study of sequentially randomized trials. These trials present as very difficult to implement for researchers due to their complicated design, but these methods make them much simpler to study and draw results from. Although not as widely used as traditional clinically randomized trials, this type of trial is likely to become more widespread in the future, requiring the use of more complex analysis techniques, such as the two discussed herein.

5.4 Extensions

This thesis focuses on the framework of medical research in order to contextualize the methods; however, these methods can be applied to data from any field. It would likely be of particular interest in the social sciences, finance, and other fields of science. Limited only by the assumptions, this method can be applied to large datasets to estimate causal effects, leading to the advancement of research across many fields. With further research and work on the program, these methods could be easily accessible by users from many fields.

A

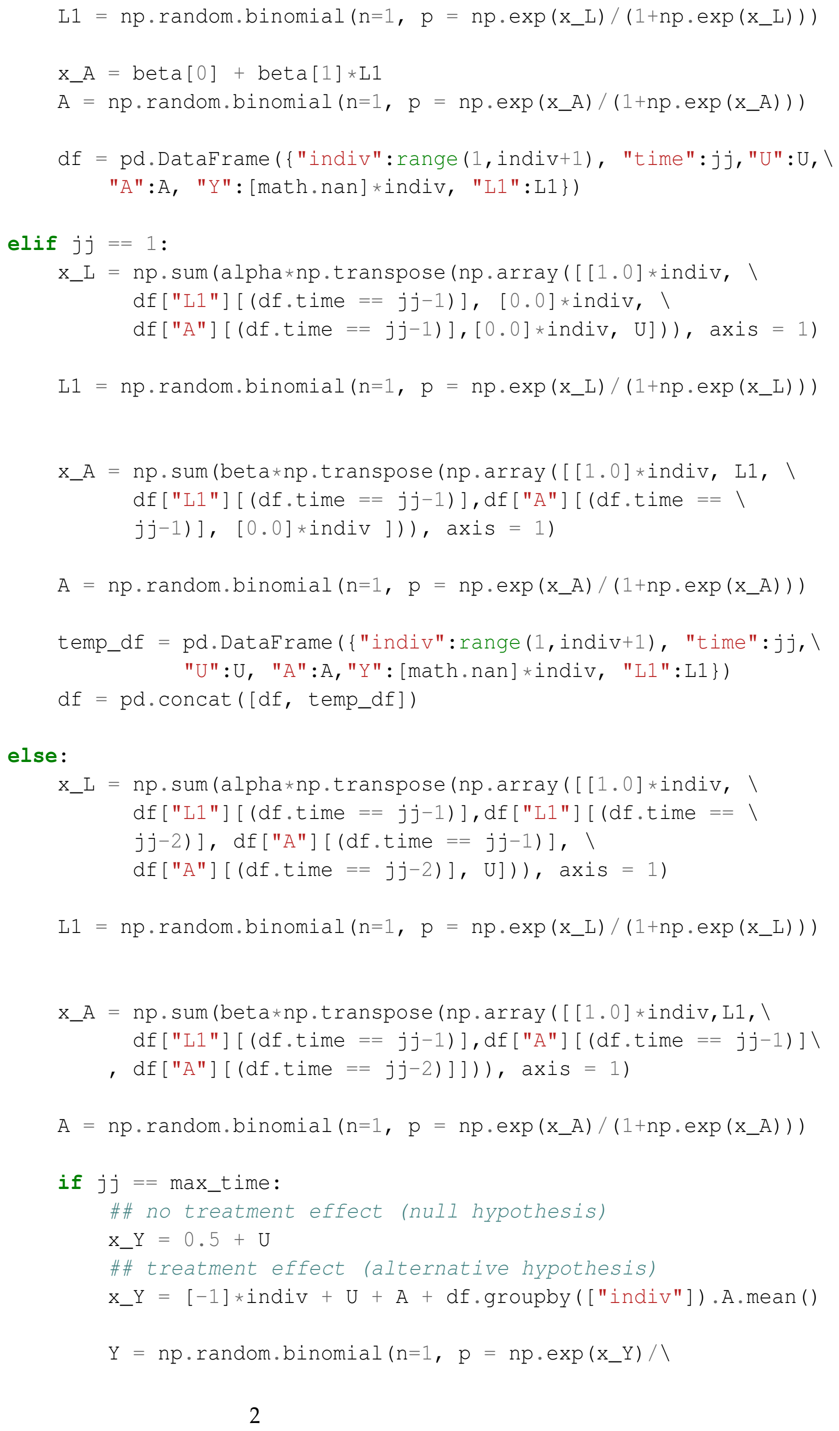
Code

The following code are the major functions written to perform the investigation here were written in Python 3. The functions follow the protocol lain out in Chapter 3 and are used according to the schematic following the code in Figure A.1.

FinalThesisFunctions

March 28, 2017

```
In [ ]: import pandas as pd
        import numpy as np
        import scipy as sp
        import sklearn as sk
        import math
        import csv
        import statsmodels.api as sm
        import statsmodels.formula.api as smf
        import random
        import matplotlib.pyplot as plt
        import pylab as plt
        import plotly.plotly as py
        import plotly
        import plotly.graph_objs as go
        from scipy.stats.stats import pearsonr
        from sklearn import linear_model, datasets
        import itertools


        ########################################################################
        ##[FUNCTION] data_creation simulates data for a given number of
        ## individuals(indiv) over a set amount of time (max_time), and can
        ## include as many covariates as desired (number_of_covariates)
        ########################################################################


        def data_creation2(indiv, max_time, number_of_covariates, Y_full, alpha,\
            beta):

            columns = ["indiv", "time","U", "A", "Y",  "L1"]
            df = pd.DataFrame(columns = columns)

            ## creating an unobserved variable that affects covariates
            U = np.random.uniform(low = 0.1, high = 1, size = indiv)

            for jj in range(0, max_time+1):
                if jj == 0:
                    x_L = alpha[0] + alpha[5]*U
```



```
    L1 = np.random.binomial(n=1, p = np.exp(x_L)/(1+np.exp(x_L)))

    x_A = beta[0] + beta[1]*L1
    A = np.random.binomial(n=1, p = np.exp(x_A)/(1+np.exp(x_A)))

    df = pd.DataFrame({"indiv":range(1,indiv+1), "time":jj,"U":U,\
        "A":A, "Y":[math.nan]*indiv, "L1":L1})

elif jj == 1:
    x_L = np.sum(alpha*np.transpose(np.array([[1.0]*indiv, \
          df["L1"][(df.time == jj-1)], [0.0]*indiv, \
          df["A"][(df.time == jj-1)],[0.0]*indiv, U])), axis = 1)

    L1 = np.random.binomial(n=1, p = np.exp(x_L)/(1+np.exp(x_L)))


    x_A = np.sum(beta*np.transpose(np.array([[1.0]*indiv, L1, \
          df["L1"][(df.time == jj-1)],df["A"][(df.time == \
          jj-1)], [0.0]*indiv ])), axis = 1)

    A = np.random.binomial(n=1, p = np.exp(x_A)/(1+np.exp(x_A)))

    temp_df = pd.DataFrame({"indiv":range(1,indiv+1), "time":jj,\
              "U":U, "A":A,"Y":[math.nan]*indiv, "L1":L1})
    df = pd.concat([df, temp_df])

else:
    x_L = np.sum(alpha*np.transpose(np.array([[1.0]*indiv, \
          df["L1"][(df.time == jj-1)],df["L1"][(df.time == \
          jj-2)], df["A"][(df.time == jj-1)], \
          df["A"][(df.time == jj-2)], U])), axis = 1)

    L1 = np.random.binomial(n=1, p = np.exp(x_L)/(1+np.exp(x_L)))


    x_A = np.sum(beta*np.transpose(np.array([[1.0]*indiv,L1,\
          df["L1"][(df.time == jj-1)],df["A"][(df.time == jj-1)]\
        , df["A"][(df.time == jj-2)]])), axis = 1)

    A = np.random.binomial(n=1, p = np.exp(x_A)/(1+np.exp(x_A)))

    if jj == max_time:
        ## no treatment effect (null hypothesis)
        x_Y = 0.5 + U
        ## treatment effect (alternative hypothesis)
        x_Y = [-1]*indiv + U + A + df.groupby(["indiv"]).A.mean()

        Y = np.random.binomial(n=1, p = np.exp(x_Y)/\
                    (1+np.exp(x_Y)))
                temp_df = pd.DataFrame({"indiv":range(1,indiv+1), \
                          "time":jj,"U":U, "A":A,"Y":Y, "L1":L1})
                df = pd.concat([df, temp_df])


            else:
                temp_df = pd.DataFrame({"indiv":range(1,indiv+1), \
                          "time":jj,"U":U, "A":A,"Y":[math.nan]*\
                          indiv, "L1":L1})
                df = pd.concat([df, temp_df])


    # creating shifted values
    if Y_full == True:
        for kk in range(1,max_time+1):
            df["L1_"+str(kk)] = df.L1.shift(kk)
            df["A_"+str(kk)] = df.A.shift(kk)
    else:
        for kk in range(1,4):
            df["L1_"+str(kk)] = df.L1.shift(kk)
            df["A_"+str(kk)] = df.A.shift(kk)

    df.sort_values(by=['time', 'indiv'], ascending=[True, True])

    return(df);

########################################################################
##[FUNCTION] Y_model_creation creates the linear regression model for
## the observed Ys based on the treatments (A) and covariates (L)
########################################################################

def Y_model_creation(df, max_time):
    temp_df = df[df.time == max_time]
    train_columns = list(df)[0:2]+list(df)[6:]
    temp_df = temp_df.astype(float)
    Y_model = sm.Logit(np.asarray(temp_df["Y"]), \
              np.asarray(sm.add_constant(temp_df[train_columns]))).fit();
    return(Y_model)


########################################################################
##[FUNCTION] covariate_model_creation creates the logistic regression
## for the observed covariate (L) data from the previous covariates
## and the previous treatments (A)
########################################################################
```

```
def covariate_model_creation(df, max_time):
    train_columns = ["L1_1", "L1_2", "L1_3", "A_1", "A_2",  "A_3"]
    L1_model = {}
    poly = PolynomialFeatures(1)

    for ii in range(1, (max_time+1)):
        temp_df = df[df.time == ii]
        if ii == 1:
            x = temp_df[["L1_1", "A_1"]]
        elif ii == 2:
            x = temp_df[["L1_1", "L1_2", "A_1", "A_2"]]
        else:
            x = temp_df[train_columns]
        L1_model[ii] = sm.Logit(np.asarray(temp_df["L1"]), \
                       poly.fit_transform(x)).fit();
    return(L1_model)


#########################################################################
##[FUNCTION] treatment_model_creation creates the logistic regression
## for the observed treatment (A) data from the current and previous
## covariates and the previous treatments (A)
#########################################################################


def treatment_model_creation(df, max_time):
    train_columns = ["L1", "L1_1", "L1_2", "A_1", "A_2", "A_3"]
    A_model = {}
    poly = PolynomialFeatures(1)

    for ii in range(0, (max_time+1)):
        temp_df = df[df.time == ii]
        if ii == 0:
            x = temp_df[["L1"]]
            A_model[ii] = sm.Logit(np.asarray(temp_df["A"]), sm.add_\
                          constant(x, has_constant = "add")).fit()
        elif ii == 1:
            x = temp_df[["L1", "L1_1", "A_1"]]
            A_model[ii] = sm.Logit(np.asarray(temp_df["A"]), poly.fit\
                          _transform(x)).fit()
        elif ii == 2:
            x = temp_df[["L1", "L1_1", "L1_2", "A_1", "A_2"]]
            A_model[ii] = sm.Logit(np.asarray(temp_df["A"]), poly.fit\
                          _transform(x)).fit()
        else:
            x = temp_df[train_columns]
            A_model[ii] = sm.Logit(np.asarray(temp_df["A"]), poly.fit\
                          _transform(x)).fit()
    return(A_model)


########################################################################
##[FUNCTION] simulation_run calculates the causal effect over an
## established number of Monte Carlo repetitions (10,000)
## using the models for outcome (Y) and the covariates (L)
########################################################################


def simulation_run(df, Y_model, L1_model_df, max_time, Y_full, \
    test_value):

    reps = 10000
    final_results = np.empty(reps)

    L_model = covariate_model_creation(df, max_time)
    poly = PolynomialFeatures(1)

    ### establishing treatment of interest
    A_test = [test_value]*(max_time+1)

    values = pd.DataFrame(np.random.choice(np.array(df["L1"][df["time"]\
             == 0]), reps))
    prod = np.empty(reps)

    prod[np.where(values[0] == 0)] = 1-np.mean(list(df["L1"][df["time"]\
                                     == 0]))
    prod[np.where(values[0] != 0)] = np.mean(list(df["L1"][df["time"]\
                                     == 0]))

    x = np.transpose(np.array([list(values[0]),[A_test[0]]*reps]))
    values[1] = L_model[1].predict(poly.fit_transform(x))

    p_v = sp.special.expit(values[1])
    values[1] = np.random.binomial(n=1, p = p_v)
    prod = prod*p_v

    x = np.transpose(np.array([list(values[1]),list(values[0]), \
        [A_test[1]]*reps, [A_test[0]]*reps]))
    values[2] = L_model[2].predict(poly.fit_transform(x))
    p_v = sp.special.expit(values[2])
    values[2] = np.random.binomial(n=1, p=p_v)
    prod = prod*p_v



    for jj in range(3, max_time+1):
        x = np.transpose(np.array([list(values[jj-1]),\
            list(values[jj-2]), list(values[jj-3]), [A_test[jj-1]]*reps,\
            [A_test[jj-2]]*reps, [A_test[jj-3]]*reps]))
        values[jj] = L_model[jj].predict(poly.fit_transform(x))

        p_v = sp.special.expit(values[jj])
        values[jj] = np.random.binomial(n=1, p=p_v)
        prod = prod*p_v

    if Y_full == "TRUE":
        Y_A = [A_test]*reps
        Y_L = np.array(values)
        Y_exp = np.array(Y_model.params[0])*([1.0]*reps) + np.sum(Y_A\
                *np.array([Y_model.params[i] for i in [1,4,6,8,10,12,\
                14,16,18,20,22,24]]), axis = 1)+np.sum([Y_model.params\
                [i] for i in [2,3,5,7,9,11,13,15,17,19,21,23]]*Y_L, \
                axis = 1)
        Y_exp = sp.special.expit(Y_exp)

    else:
        Y_A = [A_test*4]*reps
        Y_L = np.array([values[0], values[1], values[2], values[3], \
              values[4]])
        Y_exp = np.array(Y_model.params[0])*([1.0]*reps) + np.sum(Y_A\
                *np.array([Y_model.params[i] for i in [1,4,6,8]]), \
                axis = 1)+np.sum([Y_model.params[i] for i in [2,3,5,\
                7]]*Y_L, axis = 1)
        Y_exp = (np.exp(Y_exp)/(1+np.exp(Y_exp)))

    return(np.mean(prod*Y_exp))



########################################################################
##[FUNCTION] natural_course_test creates a second dataset from the
## models (L and Y) used in the g-formula to test their
## accuracy at modeling the underlying data (input df)
########################################################################
def natural_course_test(df):
    max_time = 11
    indiv = 10000
    results_mean_df = pd.DataFrame(columns = list(df))
    results_var_df = pd.DataFrame(columns = list(df))
    Y_model = Y_model_creation(df, max_time)
    L_model = covariate_model_creation(df, max_time)
    A_model = treatment_model_creation(df, max_time)
    poly = PolynomialFeatures(1)
    poly2 = PolynomialFeatures(1)
```

```
new_df = pd.DataFrame(columns = ["indiv", "time", "A", "Y",  "L1"])
for ii in range(0, max_time+1):
    if ii == 0:
        L = np.random.choice(np.array(df["L1"][df["time"] == 0]), \
            indiv)

        A = A_model[ii].predict(sm.add_constant(L, has_constant=\
            'add'))

        temp_df = pd.DataFrame({"indiv": range(0, indiv), "time": \
                  [0.0]*indiv, "A": A, "Y": [float('nan')]*indiv, \
                  "L1":L})
        new_df = pd.concat([new_df, temp_df])

    elif ii == 1:
        y = np.transpose(np.array([new_df[new_df["time"] == 0].L1,\
            new_df[new_df["time"] == 0].A]))
        L = L_model[ii].predict(poly2.fit_transform(y))

        x = np.transpose(np.array([L, new_df[new_df["time"] == 0].L1\
            ,new_df[new_df["time"] == 0].A]))
        A = A_model[ii].predict(poly.fit_transform(x))

        temp_df = pd.DataFrame({"indiv": range(0, indiv), "time": \
                  [ii]*indiv, "A": A, "Y": [float('nan')]*indiv, \
                  "L1":L})
        new_df = pd.concat([new_df, temp_df])

    elif ii == 2:

        y = np.transpose(np.array([new_df[new_df["time"] == ii-1].L1,\
            new_df[new_df["time"] == ii-2].L1, new_df[new_df["time"]\
            == ii-1].A, new_df[new_df["time"] == ii-2].A]))
        L = L_model[ii].predict(poly2.fit_transform(y))

        x = np.transpose(np.array([L, new_df[new_df["time"] == ii-1]\
            .L1,new_df[new_df["time"] == ii-2].L1,new_df[new_df\
            ["time"] == ii-1].A,new_df[new_df["time"] == ii-2].A]))
        A = A_model[ii].predict(poly.fit_transform(x))
        temp_df = pd.DataFrame({"indiv": range(0, indiv), "time": \
                  [ii]*indiv,"A": A, "Y": [float('nan')]*indiv, \
                  "L1":L})
        new_df = pd.concat([new_df, temp_df])

    else:

        y = np.transpose(np.array([new_df[new_df["time"] == ii-1].L1,\
                new_df[new_df["time"] == ii-2].L1, new_df[new_df["time"]\
                == ii-3].L1, new_df[new_df["time"] == ii-1].A, new_df[new\
                _df["time"] == ii-2].A, new_df[new_df["time"] == ii-3]\
                .A]))

            L = L_model[ii].predict(poly2.fit_transform(y))

            x = np.transpose(np.array([L, new_df[new_df["time"] == ii-1].\
                L1, new_df[new_df["time"] == ii-2].L1,\
                new_df[new_df["time"] == ii-1].A, new_df[new_df["time"]==\
                ii-2].A, new_df[new_df["time"] == ii-3].A]))
            A = A_model[ii].predict(poly.fit_transform(x))

            temp_df = pd.DataFrame({"indiv": range(0, indiv), "time": \
                      [ii]*indiv, "A": A, "Y": [float('nan')]*indiv,  \
                      "L1":L})
            new_df = pd.concat([new_df, temp_df])
    for kk in range(1,max_time+1):
        new_df["L1_"+str(kk)] = new_df.L1.shift(kk)
        new_df["A_"+str(kk)] = new_df.A.shift(kk)

    small_df = new_df[new_df["time"] == 11.0]
    cols = ['Y']+ ["time"] + ["indiv"]  + [col for col in small_df if \
           col not in  ['Y', "time", "indiv"]]
    small_df = small_df[cols]
    p_Y = np.sum(Y_model.params*sm.add_constant(small_df.ix[:,3:]), \
          axis = 1)
    new_df.Y[new_df["time"] == 11.0]= np.random.binomial(n=1, p = \
                                      sp.special.expit(p_Y)).astype(int)

    return(new_df)



#########################################################################
##[FUNCTION] pi_function creates the w_m function given the following:
## the alpha model of A_{m,i}, the dataframe, the time (m), and an
## indicator of whether this is the correct or incorrect model
#########################################################################

def pi_function(m, alpha_model, df, indiv, alpha_wrong):
    product = [1]*indiv
    for jj in range(3, m+1):
        if alpha_wrong[jj] == False:
            x = alpha_model[jj].predict(sm.add_constant(df[df.time ==\
                jj][["L1", "L1_1", "L1_2", "A_1", "A_2"]], \
                has_constant='add'))
        else:
            x = alpha_model[jj].predict(sm.add_constant(df[df.time ==\
                jj][["L1_3","A_3"]], has_constant='add'))
        product = product*x

    x = np.array(np.divide([1]*indiv, product))
    x[np.where(df[df.time == m]["A_1"] == 0.0)] = 1 - x[np.where(df\
            [df.time == m]["A_1"] == 0.0)]
    return(x)

########################################################################
##[FUNCTION] alpha_model_creation creates the logistic regression
## for the observed treatment (A) data from the current and previous
## covariates and the previous treatments (A) over all time periods and
## individuals
########################################################################


def alpha_model_creation(df, wrong):
    temp_df = df[df["time"]>2.0]
    if wrong == True:
        alpha_model = sm.Logit(np.asarray(temp_df.A),np.asarray(sm.add\
                      _constant(temp_df[["L1_3", "A_3"]], has_constant\
                      ='add'))).fit()

    else:
        alpha_model = sm.Logit(np.asarray(temp_df.A),np.asarray(sm.add\
                      _constant(temp_df[["L1", "L1_1", "L1_2", "A_1", \
                      "A_2"]], has_constant='add'))).fit()

    return(alpha_model)


########################################################################
##[FUNCTION] DR_estimate_creation calculates the causal effect for a
## given treatment of interest (test_value), including an indicator
## of whether the correct or incorrect model is being used
########################################################################

def DR_estimate_creation_bin_time(test_value, max_time, df, indiv, \
    wrong_alpha_model, wrong_s_model, alpha_model, int_term):

    A_test = [test_value]*indiv
    model_df = pd.DataFrame(columns = ["time", "beta_0", "beta_1",
               "beta_2","beta_3", "beta_4", "beta_5", "beta_6", "phi"])
    time_counter = max_time+1
    T = df[df.time == max_time]["Y"]

    poly = sk.preprocessing.PolynomialFeatures(interaction_only = True)
```

```
while(time_counter > 3.0):
    time_df = df.loc[df.time == time_counter-1]
    pi = pi_function(time_counter-1, alpha_model, df, indiv, \
         wrong_alpha_model)
    time_df["pi"] = pi
    if wrong_s_model[time_counter-1] == True:
        train_columns = list(time_df)[0:2] + list(time_df)[12:14]\
        +["pi"]
        reg_columns = '+'.join(map(str, np.append(list(time_df)\
                    [0:2],np.append(list(time_df)[12:14], ["pi"]))))
    else:
        train_columns = list(time_df)[0:2] + list(time_df)[6:10]+\
                        ["pi"]
        if int_term == True:
            x = list(itertools.combinations(np.append(list(time_df)\
                [0:2], list(time_df)[6:10]), 2))
            y = ['*'.join(map(str, np.array([x[i][0], x[i][1]]))) \
                 for i in range(len(x))]
            z = '+'.join(map(str, y))
            reg_mid_columns = '+'.join(map(str, np.append(list(\
                              time_df)[0:2], np.append(list(time_df)\
                              [6:10],["pi"]))))
            reg_columns = '+'.join(map(str, np.array([reg_mid_
                          columns,z])))
        else:
            reg_columns = '+'.join(map(str, np.append(list(time_df)\
                          [0:2],np.append(list(time_df)[6:10],["pi"])))
    time_df = time_df.astype(float)

    formula = "T~"+reg_columns
    glm_model = smf.glm(formula = formula, data = time_df, family=\
                sm.families.Binomial(link=sm.families.links.logit))
    try:
         glm_results = glm_model.fit()
    except Exception as ex:
        return(float("nan"), float("nan"))

    pi2 = pi_function(time_counter-2, alpha_model, df, indiv, \
          wrong_alpha_model)

    time_df["A"] = np.array(A_test)

    if test_value == 1:
        if wrong_alpha_model[time_counter-1] == True:
            pi2 = pi2*alpha_model[time_counter-1].predict(\
                  sm.add_constant(time_df[["L1_3", "A_3"]], \
                  has_constant = "add"))
        else:
            pi2 = pi2*alpha_model[time_counter-1].predict(\
                sm.add_constant(time_df[["L1", "L1_1", "L1_2", \
                "A_1", "A_2"]], has_constant = "add"))

    elif test_value == 0:
        if wrong_alpha_model[time_counter-1] == True:
            pi2 = pi2*(1-alpha_model[time_counter-1].predict(\
                  sm.add_constant(time_df[["L1_3", "A_3"]], \
                  has_constant = "add")))
        else:
            pi2 = pi2*(1-alpha_model[time_counter-1].predict(\
                  sm.add_constant(time_df[["L1", "L1_1", "L1_2",\
                  "A_1", "A_2"]], has_constant = "add")))


    time_df["pi"] = pi2
    T = glm_results.predict(time_df[train_columns])
    time_counter = time_counter-1

values = np.array([np.mean(df.Y), np.mean(df.A), np.mean(df.L1), \
         np.mean(df.U), pearsonr(df.Y[df.time == 11], \
         df.A[df.time == 11])[0], pearsonr(df.Y[df.time == 11], \
         df.L1[df.time == 11])[0], pearsonr(df.Y[df.time == 11], \
         df.U[df.time == 11])[0], pearsonr(df.A, df.L1)[0], \
         pearsonr(df.U, df.L1)[0], pearsonr(df.A, df.U)[0]])
return(np.nanmean(T), values)
```

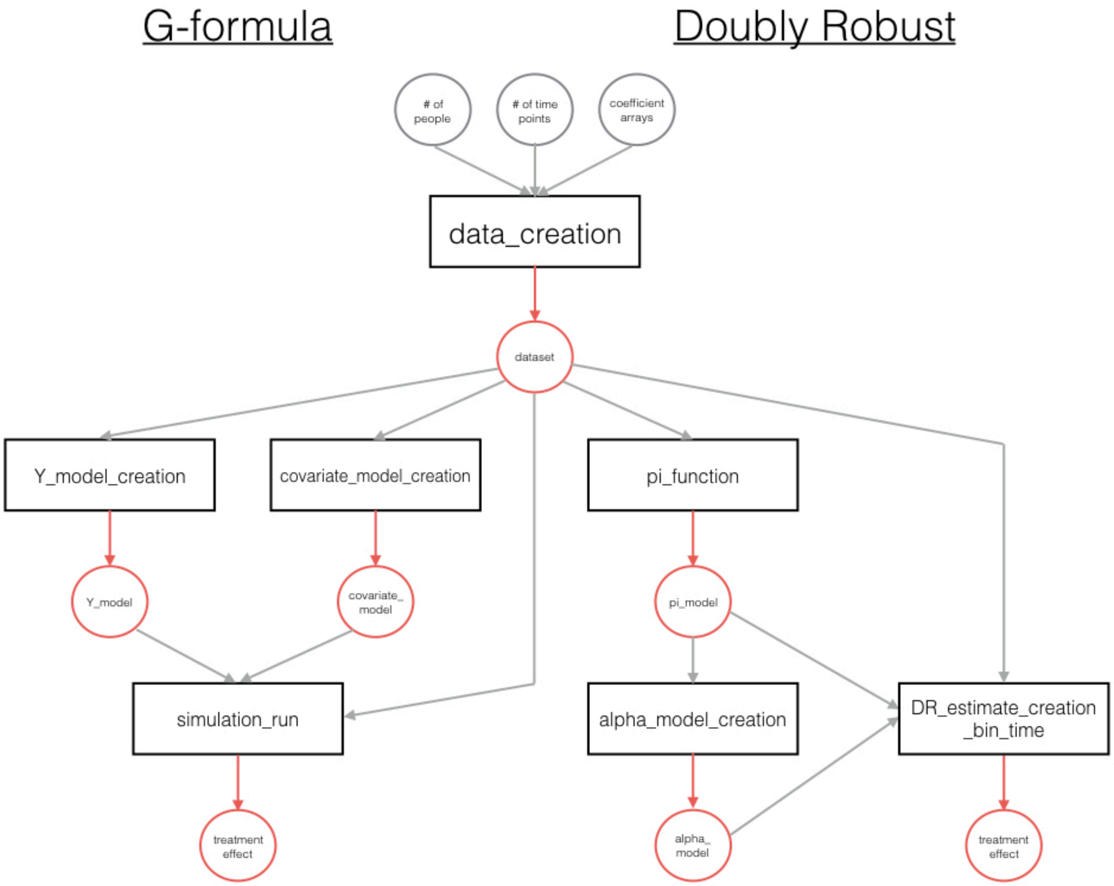


Figure A.1: This schematic shows how the above functions in Python can be utilized in relation to each other in order to execute the two methods. Black rectangles represent functions. Circles represent inputs/outputs, red showing function outputs and grey showing user inputs.

B

Extra Results

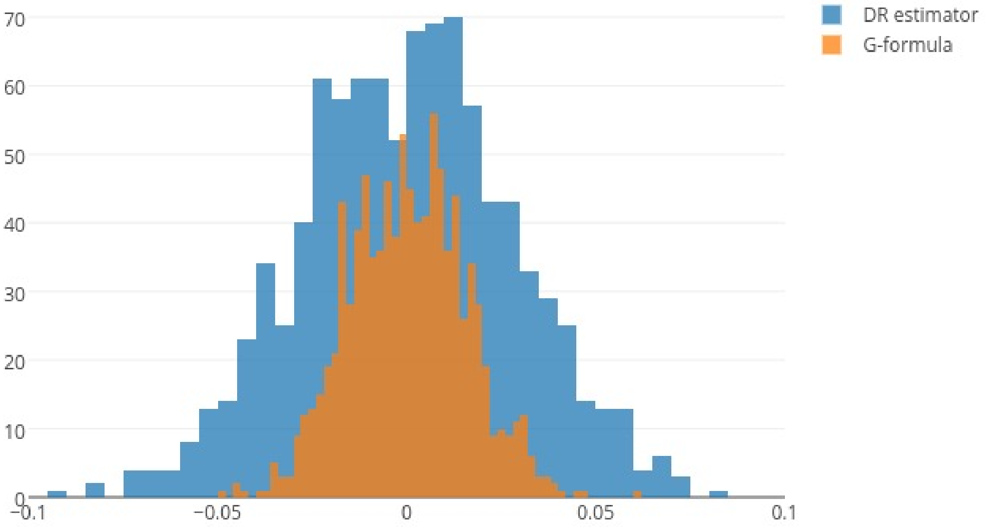


Figure B.1: A histogram showing the results of the 1,000 simulations. The y-axis shows frequency of value and the x-axis shows the causal treatment effect estimates.

Models Correctly Specified	Average Bias of Estimate	Std. Error of Bias
$\pi_3, \ldots, \pi_K$	0.026	0.005
$\pi_3, \ldots, \pi_{K-1}, s_K$	0.024	0.004
$\pi_3, \ldots, \pi_{K-2}, s_{K-1}, s_K$	0.023	0.004
$\pi_3, \ldots, \pi_{K-3}, s_{K-2}, s_{K-1}, s_K$	0.023	0.004
$\pi_3, \ldots, \pi_{K-4}, s_{K-3}, \ldots s_K$	0.023	0.004
$\pi_3, \ldots, \pi_{K-5}, s_{K-4}, \ldots s_K$	0.023	0.004
$\pi_3, \ldots, \pi_{K-6}, s_{K-5}, \ldots s_K$	0.023	0.004
$\pi_3, \ldots, \pi_{K-7}, s_{K-6}, \ldots s_K$	0.024	0.004
$\pi_3, s_4, \ldots s_K$	0.025	0.004
$s_3, \ldots s_K$	0.026	0.044
$s_3, \ldots s_{10}, \pi_K$	0.027	0.025
$s_3, \ldots s_9, \pi_{10}, \pi_K$	0.028	0.017
$s_3, \ldots s_8, \pi_9, \ldots \pi_K$	0.048	0.026
$s_3, \ldots s_7, \pi_8, \ldots \pi_K$	0.060	0.025
$s_3, \ldots s_6, \pi_7, \ldots \pi_K$	0.064	0.021
$s_3, s_4, s_5, \pi_6, \ldots \pi_K$	0.052	0.005
$s_3, s_4, \pi_5, \ldots \pi_K$	0.080	0.005
$s_3, \pi_4, \ldots \pi_K$	0.196	0.005
$s_3, \pi_4, s_5, \pi_6, s_7, \pi_8, s_9, \pi_{10}, s_{11}$	0.031	0.005
$\pi_3, s_4, \pi_5, s_6, \pi_7, s_8, \pi_9, s_{10}, \pi_{11}$	0.034	0.005

Table B.1: Further results of the testing of multiple robustness of the doubly robust method as described in Section 4.4.

References

[1] Bang, H. & Robins, J. M. (2005). Doubly robust estimation in missing data and causal inference models. Biometrics, 61(4), 962–973.

[2] Bembom, O. & van der Laan, M. J. (2007). Statistical methods for analyzing sequentially randomized trials. JNCI: Journal of the National Cancer Institute, 99(21), 1577.

[3] Cole, S. R. & Frangakis, C. E. (2009). The consistency statement in causal inference: a definition or an assumption? Epidemiology, 20(1), 3–5.

[4] Drugs.com (2017). Fda drug approval process.

[5] Dzau, V. J., McClellan, M. B., McGinnis, J. M., Burke, S. P., Coye, M. J., Diaz, A., Daschle, T. A., Frist, W. H., Gaines, M., Hamburg, M. A., et al. (2017). Vital directions for health and health care: Priorities from a national academy of medicine initiative. JAMA.

[6] Edwards, A. (2005). Ra fisher, statistical methods for research workers, (1925). Landmark Writings in Western Mathematics 1640–1940, (pp. 856).

[7] Fisher, R. A. (1935). The design of experiments. 1935. Oliver and Boyd, Edinburgh.

[8] Fitzmaurice, G., Davidian, M., Verbeke, G., & Molenberghs, G. (2008). Longitudinal data analysis. CRC Press.

[9] Hernán, M. A. & Robins, J. M. (2006). Estimating causal effects from epidemiological data. Journal of epidemiology and community health, 60(7), 578–586.

[10] Hernan, M. A. & Robins, J. M. (2016). Causal Inference. Chapman & Hall/CRC.

[11] Imbens, G. W. & Rubin, D. B. (2015). Causal inference in statistics, social, and biomedical sciences. Cambridge University Press.

[12] Lodi, S., Phillips, A., Logan, R., Olson, A., Costagliola, D., Abgrall, S., van Sighem, A., Reiss, P., Miró, J. M., Ferrer, E., et al. (2015). Comparative effectiveness of immediate antiretroviral therapy versus cd4-based initiation in hiv-positive individuals in high-income countries: observational cohort study. The Lancet HIV, 2(8), e335–e343.

[13] Pearl, J. & Robins, J. (1995). Probabilistic evaluation of sequential plans from causal models with hidden variables. In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence (pp. 444–453).: Morgan Kaufmann Publishers Inc.

[14] Robins, J. (1986). A new approach to causal inference in mortality studies with a sustained exposure period?application to control of the healthy worker survivor effect. Mathematical Modelling, 7(9-12), 1393–1512.

[15] Rosenbaum, P. R. & Rubin, D. B. (1984). Reducing bias in observational studies using subclassification on the propensity score. Journal of the American statistical Association, 79(387), 516–524.

[16] Rowan, A. (2011). Avoiding animal testing. The Scientist.

[17] Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. Journal of educational Psychology, 66(5), 688.

[18] Rubin, D. B. (1978). Bayesian inference for causal effects: The role of randomization. The Annals of statistics, (pp. 34–58).

[19] Rubin, D. B. (1980). Randomization analysis of experimental data: The fisher randomization test comment. Journal of the American Statistical Association, 75(371), 591–593.

[20] Rubin, D. B. (1984). William g. cochran?s contributions to the design, analysis, and evaluation of observational studies. WG Cochran's impact on statistics, (pp. 37–69).

[21] Splawa-Neyman, J., Dabrowska, D. M., Speed, T. P., et al. (1990). On the application of probability theory to agricultural experiments. essay on principles. section 9. Statistical Science, 5(4), 465–472.

[22] VanderWeele, T. J. (2009). Concerning the consistency assumption in causal inference. Epidemiology, 20(6), 880–883.

[23] VanderWeele, T. J. & Arah, O. A. (2011). Unmeasured confounding for general outcomes, treatments, and confounders: bias formulas for sensitivity analysis. Epidemiology (Cambridge, Mass.), 22(1), 42.

[24] Wright, J. D. (2015). International encyclopedia of the social and behavioral sciences.

[25] Young, J. G., Cain, L. E., Robins, J. M., O'Reilly, E. J., & Hernán, M. A. (2011). Comparative effectiveness of dynamic treatment regimes: an application of the parametric g-formula. Statistics in biosciences, 3(1), 119–143.