An Investigation of Methods of Causal Inference: The Development of a Multiply Robust Estimator

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 advisor: Professors James Robbins and Miguel Hernan Morgan F. Breitmeyer

An Investigation of Methods of Causal Inference: The Development of a Multiply Robust Estimator

Abstract

The statistical field of causal inferences seeks to develop methods which are able to prove causation from observational data without the need for costly and time intensive randomized trials. Two such methods, the g-formula and doubly robust estimation, are investigated and compared in this thesis. The g-formula is computationally intensive and has the potential to be biased by the incorrect specification of models. However, with care on the part of the researcher and a high number of Monte Carlo simulations, a precise measurement of causal treatment effect can be gained. Furthermore, the doubly robust method is stronger at detecting underlying correlation between outcome and treatment status than the g-formula. On the other hand, the doubly robust is slightly less precise in its estimation of causal treatment effect, with slightly higher variance, but 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. This developed method is not only multiply robust, but its implementation in Python is also significantly more computationally efficient. Furthermore, it is decreasing many barriers to use for the scientific community, opening pathways to countless imminent discoveries from the massive data currently available and underutilized.

Contents

This is the dedication.

Acknowledgments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi commodo, ipsum sed pharetra gravida, orci magna rhoncus neque, id pulvinar odio lorem non turpis. Nullam sit amet enim. Suspendisse id velit vitae ligula volutpat condimentum. Aliquam erat volutpat. Sed quis velit. Nulla facilisi. Nulla libero. Vivamus pharetra posuere sapien. Nam consectetuer. Sed aliquam, nunc eget euismod ullamcorper, lectus nunc ullamcorper orci, fermentum bibendum enim nibh eget ipsum. Donec porttitor ligula eu dolor. Maecenas vitae nulla consequat libero cursus venenatis. Nam magna enim, accumsan eu, blandit sed, blandit a, eros.

1

Introduction

Causation versus correlation - the age old debate continues on among statisticians, scientists, and students alike. Correlation is the easier idea to understand here: 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 quite simple too. Find a group of people who do drink red wine and eat dark chocolate and a group of people who do not, and compare their resting heart rates and HDL ("good" cholesterol) levels. If the wine drinkers and chocolate consumers have better heart health, then it can be said that consuming these is associated with a healthier heart. As lovely as that sounds, there is a catch. This method only demonstrates that consuming red wine and chocolate is correlated with a healthy heart, but it does not prove that red wine and chocolate actually cause the healthy heart. The distinction being that perhaps everyone who consumes red wine and dark chocolate could also exercise frequently, be of a healthy weight, or be younger than those who do not, all possible factors that could lead to a healthier heart separate from an individual's red wine drinking and chocolate eating habits.

Yet, this begs the question of how to actually prove that the red wine and dark chocolate caused healthier hearts. Answering this question is not quite as simple as showing correlation, and traditionally, a fully randomized blinded experiment was required. A randomized trial experiment involves enrolling a sufficient number of patients, who are randomly categorized into one of two groups: those who are told to drink red wine (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 a couple of months or even years. The individuals in both groups would have their heart rate and their HDL levels measured along the way. After many months of this, despite several individuals who surely had given up and dropped out of the study, data is acquired at the very end. Using this data, the researchers could compare the heart health of the two groups and come to a definitive conclusion

(pending statistical significance) of whether drinking red wine caused a healthier heart. This process is complicated, costly, and until recently, the only option to demonstrate causation.

This process is commonly used in medicine to test the efficacy of a treatment drug. On average, it takes 12 years and more than $350 million USD to get a new drug through clinical trial testing and FDA approval.[*] FDA approval requires the proof of statistically significant sufficient causation of the drug's efficacy through a randomized trial. The difficulty surrounding these trials makes the medical innovation process slow, expensive, and inefficient, purely because this means of proving causation is so tedious.

However, in the last few decades, a field of statistics known as causal inference has emerged, which studies and creates various methods to attempt to prove causation by better means than the randomized trial. This field emphasizes methods that can work strictly from observational data, meaning data that is already collected and does not need to be controlled or dictated by the investigator. Some examples of observational data include hospital data accumulated over millions of patients and across years, tracking symptoms, various levels, and outcomes, demographic surveys, and data from other experimental trials which is now being reconsidered for a different question of interest. Using these alternative methods, drug efficacy could be proven with significantly less cost and time, significantly impacting millions of lives.

This thesis is a study of two such methods of causal inference: the g-formula and a doubly robust estimator. Both methods seek to prove the same causal effect as a randomized trial from purely observational data. The doubly robust estimator is an improved development on the g-formula and is

[*]New Drug Approval Process, https://www.drugs.com/fda-approval-process.html

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, leading to a better estimator.

2

Background

A TRADITIONAL UNDERSTANDING OF CAUSATION comes from the field of medicine, where researchers can perform a controlled experiment to prove causation. This type of study contains two sample groups, one which receives no treatment (the placebo group) and one which receives the treatment (the treatment group). Individuals are randomly allocated into one group, and by com-

paring the outcome of these two groups, the researchers can demonstrate whether the outcome for patients receiving treatment differs significantly from the controls. By quantifying the difference in outcomes between the groups, researchers can demonstrate an association between treatment and outcome. However, because of the randomized nature of the trial, association is causation.[*] [4]

2.1 Causal Effects

To translate this idea 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 the individual $i \in \{0, 1, 2, \ldots n\}$ with n the total number of individuals.[†] The random variable 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 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, the desired value is the difference in Y under the varying conditions of A. Notationally, this is the difference between $Y^{a=1}$ [‡], the outcome that would be observed under treatment, and $Y^{a=0}$, the outcome that would be observed under no treatment. This is in

[*] This idea is discussed further in Section 2.3.

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

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

comparison to the observed outcome of Y or Y^A.

A causal effect can be seen on an individual level if $Y_i^{a=1} \neq Y_i^{a=0}$. 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

$$Y_i^{a=1} - Y_i^{a=0} \tag{2.1}$$

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 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.[5] 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. Furthermore, if patients ina 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 $\overline{A} = \{A_0 = 1, A_1 = 1, \ldots, A_K = 1\} = \{1, 1, \ldots 1\}$, which can also be written as $\overline{A} = \overline{1}$. 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 Regimes

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, $\bar{L}_k$ and $\bar{A}_{k-1}$. 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

A specific type of deterministic treatment regime is the sequentially randomized trial, 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 $\bar{A}_k$ in time period k.[12] 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.

2.2 Confounding

The use of the covariate $\bar{L}$ is a measurable proxy for an unmeasured and unknown underlying confounder, U. Theoretically, U should directly impact both L and Y, but not A, so it indicates a backdoor path between A and Y through U.[11] 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. By eliminating the effect of U, there will no bias in the estimate of causal effect. The methods for doing so will be discussed in Sections 2.4 and 2.5.

In this scenario, L is referred to as a confounder for the effect of A, reflective of the fact that the underlying bias was the unknown of U and L is being used to account for that. It can be shown that

in order to validly estimate the joint effect of all A_k simultaneously and without bias, it is sufficient (but not necessary) to block all backdoor paths from U to any A_k for all k.[8]

2.3 Identifiability Assumptions

It is sufficient to show that causal effects are valid and identifiable, meaning they have a single measurement of effect, on the following three assumptions: consistency, positivity, and exchangeability.[2,5] Under these three assumptions, the data closely resembles an ideal randomized trial. Through this, causation can be inferred, rather than simply association. Although the methods are directly testing association, these assumptions allow the tests to measure causation.

2.3.1 Consistency

Consistency is the idea that an individual's potential outcome and their observed outcome are equal[2,5]. Statistically, 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 presence of multiple or varying treatment options, such as different surgeons perform a procedure or even varying procedures. Protection against this is partially in the understanding and reasonable pruning of the data. This can be done through clear and precise questions of interest, and hopefully, detailed data that allows for comprehensive refinement.

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

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

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 treatment.[5] There should be no bias to either group to respond favorably or not to treatment or lack thereof; thus, the results should be equivalent if any subject is moved from one group to the other.

Statistically, this is $P[Y^a = 1 \mid A = 1] = P[Y^a = 1 \mid A = 0] = P[Y^a = 1]$. This means that Y^a is independent of A, and the treatment has no predictive power of the outcome. This independence allows for several conclusions. Firstly, $E[Y^a \mid A = a'] = E[Y^a]$ by definition of independence.

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, exchangeability is attainable when considering obesity, but is more difficult when the confounder has a high mortality rate, such as Therefore, conditional exchangeability is obtained: $P[Y^a = 1 \mid A = a, L = l] = P[Y^a = 1 \mid A \neq a, L = l]$, i.e. $Y^a \perp\!\!\!\perp A \mid L$.[5] Conditional exchangeability guarantees the ability to measure effects using complete data.

The power of a randomized trial is that it should theoretically create exchangeability. By randomly putting subjects into their groups, there should be no reason that the patients between the two groups differ or will respond to treatment differently. However, exchangeability can be ob-

tained in an observational study if $P[A_k = 1]$ depends only on $\{\overline{A}_{k-1}, \overline{L}_k\}$ and thus,

$$P[A_k \mid \overline{A}_{k-1}, \overline{L}_k] \perp\!\!\!\perp U \tag{2.7}$$

By accounting for U using L in a time-varying treatment method, it can be seen that

$$Y \perp\!\!\!\perp A_k \mid \overline{L}_k, \overline{A}_{k-1} \tag{2.8}$$

which is referred to as having no unmeasured time-varying confounders. Although guaranteed for fixed treatments, sequential exchangeability is not guaranteed.[11] Approximate exchangeability can be achieved in practice by including as many covariates as is feasibly reasonable, but this is still risky business there is no known method for computationally measuring or empirically testing sequential exchangeability. However, the assumption of conditional exchangeability is the same as for fixed treatment models and is sufficient for determining causal effect.

2.3.3 Positivity

Positivity is the condition that a specified conditional probability is well-defined, meaning that for every value of the covariate L, there exist subjects with a specified value of a.[4] Statistically, this looks like

$$P[A = a \mid L = l] > 0 \quad \forall l, \text{ such that } P[L = l] \neq 0 \tag{2.9}$$

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

$$P[A_k = a_k \mid \bar{L}_k, \bar{A}_{k-1}] > 0 \quad \forall A_k, \text{ such that } P[\bar{L}_k = \bar{l}_k, \bar{A}_{k-1} = \bar{a}_{k-1}] \neq 0 \tag{2.10}$$

2.4 IP Weighting

Many of the concerns discussed above can by addressed using the method of IP weighting by simulating a pseudo-population, in which every individual has two data inputs, the expected observed outcomes under treatment and under no treatment. The method by which this is done is by considering 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 of the possible A and L combinations

$$n \cdot P[Y = y \mid A = a, L = l] \cdot P[A = a \mid L = l] \cdot P[L = l] \cdot \frac{1}{P[A = a \mid L = l]} \tag{2.11}$$

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 prob-

ability of receiving treatment and written as,[6]

$$e(x) = \frac{N_t(x)}{N_c(x) + N_t(x)} = P[A = a \mid L = l] \tag{2.12}$$

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

This form in expression 2.11 can be used to solve for the standardized mean as follows,

$$E[Y^a] = \sum_l n \cdot P[Y = y \mid A = a, L = 1] \cdot P[A = a \mid L = l] \cdot P[L = l] \cdot \frac{1}{P[A = a \mid L = 1]} \tag{2.13}$$

$$= \sum_l n \cdot P[Y = y \mid A = a, L = 1] \cdot P[L = l] \tag{2.14}$$

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

This leads to the confounders being accounted for or eliminated in the pseudo-population. As a result, the causal effect of A on Y can effectively be 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. This allows us to estimate IP weights

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 looks like this,

$$P[Y^a = 1] = \sum_l P[Y^a = 1 \mid L = l] \cdot P[L = l] \tag{2.16}$$

$$= \sum_l P[Y = 1 \mid L = l]P[L = l] \tag{2.17}$$

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

$$E[Y^a] = E[Y \mid L = l, A = a] \cdot P[L = l] \tag{2.18}$$

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 were used 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

For the purposes of this study, a data generating algorithm was created to provide consistent and easily accessible data for many simulations. The data generated was time-varying and sequentially randomized according to the following schematic.

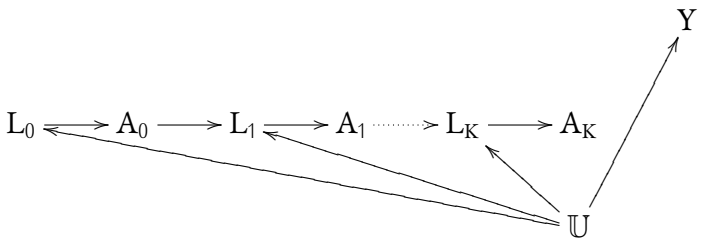

Figure 3.1: Diagram of conditional dependencies in data generating process.

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 $\{0, \ldots K = 11\}$.

1. Take the predetermined coefficients $\vec{\alpha}$ and $\vec{\beta}$, which 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 $\vec{\alpha} \sim \text{Uniform}(-1.0, 1.0)$ and $\vec{\beta} \sim \text{Uniform}(-1.0, 1.0)$. The one change made was that α_5 had 1.5 added to the randomly generated value to ensure that the underlying covariate had significant impact.

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

3. Using the following logistic expressions, 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.

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

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

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

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

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

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

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

4. Obtain a final Y_i value for each individual where $Y_i \sim \mathcal{N}(\mu = U, \sigma = 1)$

Note that the final outcome Y value is independent of A and therefore, the treatment has no impact on the outcome. This was done to ensure that the causal treatment effect would be zero and bias could be easily measured throughout the study.

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.[9,11,5] 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,

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

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 which have feedback to each other. This is equivalent to each L_k being dependent on A_{k-1}.[9] 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) will not.[11] This is because these other methods are unable to estimate the joint effect of all treatment values $\{A_0, A_1 \ldots A_K\}$ simultaneously and without bias.[3]

The g-formula method has been shown to have a smaller variance than IP weighting methods, but this comes with added parametric modeling assumptions.[12] The smaller variance is due to the

fact that the g-formula uses maximum likelihood estimates, in comparioson to the semi-parametic 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 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, A_k and L_k for all k. IP weighting methods are also dependent on the accuracy of their models, just different models such as for A_k conditional on $\overline{L}_k, \overline{A}_{k-1}$.

3.2.1 Protocol

The method is performed in several steps, as follows

1. Create outcome models: Create models for the outcome variable Y and the covariates, L_k at each time using the original dataset. The model for Y is regressed 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 $L_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}$,

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

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

   A time lag of only three historical values was deemed sufficient for the model of L_k because each of the L_k are dependent on previous time history so three historical values should be sufficient to capture that.

   Note that for initial time points where there was insufficient history for the full model,

smaller models were created as follows

$$\text{Logit}[L_1 \mid L_0, A_0] = \gamma'_0 + \gamma'_1 L_0 + \gamma'_2 A_0 \tag{3.11}$$

$$\text{Logit}[L_2 \mid L_0, L_1, A_0, A_1] = \gamma''_0 + \gamma''_1 L_{k-1} + \gamma''_2 L_{k-2} + \gamma''_3 A_{k-1} + \gamma''_4 A_{k-2} \tag{3.12}$$

2. Predict using Monte Carlo: Using the model created in step 2, predict the outcome Y for the two new datasets created in step 1, conditioned on the given A and L values.

   Using expressions 3.9 through 3.12, a Monte Carlo simulation must be performed to gain prediction values. This is because it is impractical to calculate expression 3.10 directly for a continuous L. This process is done as follows for time $k = \{0, \ldots, K\}$, and individuals $i = \{1, \ldots, n\}$ 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 $i \in \{1, \ldots, n\}$.

   (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 $\bar{L}_K$ and all the probabilities $P[L_k = l_k \mid \bar{L}_{k-1}, \bar{A}_{k-1}]$

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

   (f) Take the product of all the probabilities $P[L_k = l_k \mid \bar{L}_{k-1}, \bar{A}_{k-1}]$ for $k = 0, \ldots, K$ and $\mathbb{P}[Y \mid \bar{A}_K, \bar{L}_K]$ 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.

To do this process using non-parametric estimates, the Monte Carlo simulation is unnecessary. Instead, the methodology would be to create two new simulated datasets, the first having all individuals under no treatment ($A = 0$) and the second having all individuals treated ($A = 1$). Each of these

new datasets has the same size as the original and the same "individuals", meaning the same covariate L distribution, just changed values for A. For these two datasets, delete the outcome values for Y to treat it as a missing data point. Then, the outcome models would be calculated as above. The prediction step, however, would differ, instead using the models directly to get predicted values for each individual in the extra two datasets, rather than a Monte Carlo simulation. Finally, 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 and standardization 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.[5] IP weighting estimates $P[A = a \mid L = l]$, while standardization estimates $P[Y \mid A = a, L = l]$ and $P[\overline{L}_k = l_k \mid \overline{L}_{k-1}, \overline{A}_{k-1}]$.

Some important improvements of the method of doubly robust estimation include that it does not make use of the observed data treatment density, as the methods of IP weighting and standardization do, allowing for missing data to be properly accounted for and a lack of skewing due to over-represented populations in the 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. This is the derivation of the term doubly robust. Furthermore, however, will be evidence pointing to what order combinations of correctly and incorrectly specified models lead to the introduction of 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.

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

   This model can be rewritten as the following.

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

2. Set $\hat{T}_{K+1} = Y$
3. Recurse for $m = K + 1, \ldots, 2$
   (a) Use IRLS and a specified parametric regression model to get

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

   which gives the conditional expectation of

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

   The known function s_m is specified on a case by case basis, and in this case was chosen to be as follows for the unknown parameter β.

$$s_m(\bar{L}_m, \bar{A}_m; \beta_m) = \theta_0 + \theta_1 L_m + \theta_2 A_m + \theta_3 L_{m-1} + \theta_4 A_{m-1} + \theta_5 L_{m-2} + \theta_6 A_{m-2} \tag{3.17}$$

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

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

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

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

(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

$$0 = \tilde{\mathbb{E}}\left[\left[\hat{\Upsilon}_m - \Psi\{s_{m-1}(\overline{L}_{m-1}, \overline{A}_{m-1}; \hat{\beta}_{m-1}) + \hat{\phi}_{m-1}\overline{\pi}^{-1}_{m-1}(\hat{\alpha})\}\right]\left(\frac{\partial s(\overline{L}_{m-1}; \beta_{m-1})}{\partial \beta'_{m-1}, \overline{\pi}^{-1}_{m-1}(\hat{\alpha})}\right)\right] \tag{3.20}$$

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

(c) Set

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

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

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

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

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

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

3.4 Variance Estimate

In order to compute the variance of the estimates obtained using the above two methods, a bootstrapping simulation was conducted. 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 were implemented and tested for their efficacy at evaluating average causal treatment effect. Results and discussion are presented here.

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,*

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

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

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

Five new data sets were simulated to determine the natural course of these models. Table 4.1 below compares the true mean of the original data frame and the average across the five simulated dataframes.

The results of the natural course in Table 4.1 show that the models chosen appear to estimate the data quite well. Not only are the means close to the true mean, but the variance is low enough that our confidence intervals all cover the true mean. Therefore, it can be concluded that these models are a good choice for adequately modeling the underlying data in the g-formula.

*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.

Variable	True Mean	Natural Course Average Mean	Natural Course 95% CI
$\mathbb{E}[Y]$	0.714	0.719	(0.705, 0.733)
$\mathbb{E}[A]$	0.848	0.847	(0.840, 0.854)
$\mathbb{E}[L]$	0.833	0.811	(0.793, 0.830)

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 the models are created off of. The natural course average mean is the mean of means from the five datasets. The 95% confidence interval is centered at this mean of means and based on the five datasets.

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.

The results of these simulations can be seen below in Table 4.2 and Figure 4.1. 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 relationship between the data and the causal effect estimates was examined in Figure 4.2. Of note, the doubly robust method is able to capture correlation between Y and A which the g-formula method does not, an important indicator that it successfully picks up treatment effect. This likely contributes to the more precise causal treatment effect measure.

[†] As discussed in Section 3.1, no effect of A on Y was included in this study, 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
G-Formula	0.0033	0.016	0.00024	(-0.00063,0.00128)
Doubly Robust	0.00026	0.031	0.00079	(-0.00149, 0.00212)

Table 4.2: A table presenting the results of 1,000 data simulations, 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.

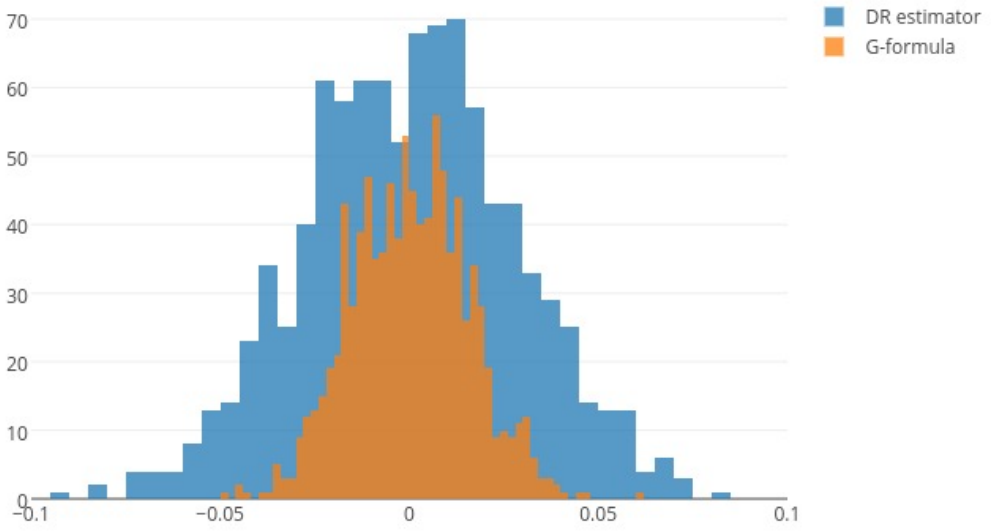

Figure 4.1: A histogram showing the results of the 1,000 simulations. The y-axis shows frequency of value and the x-axis shows the causal treatment effect value measured.

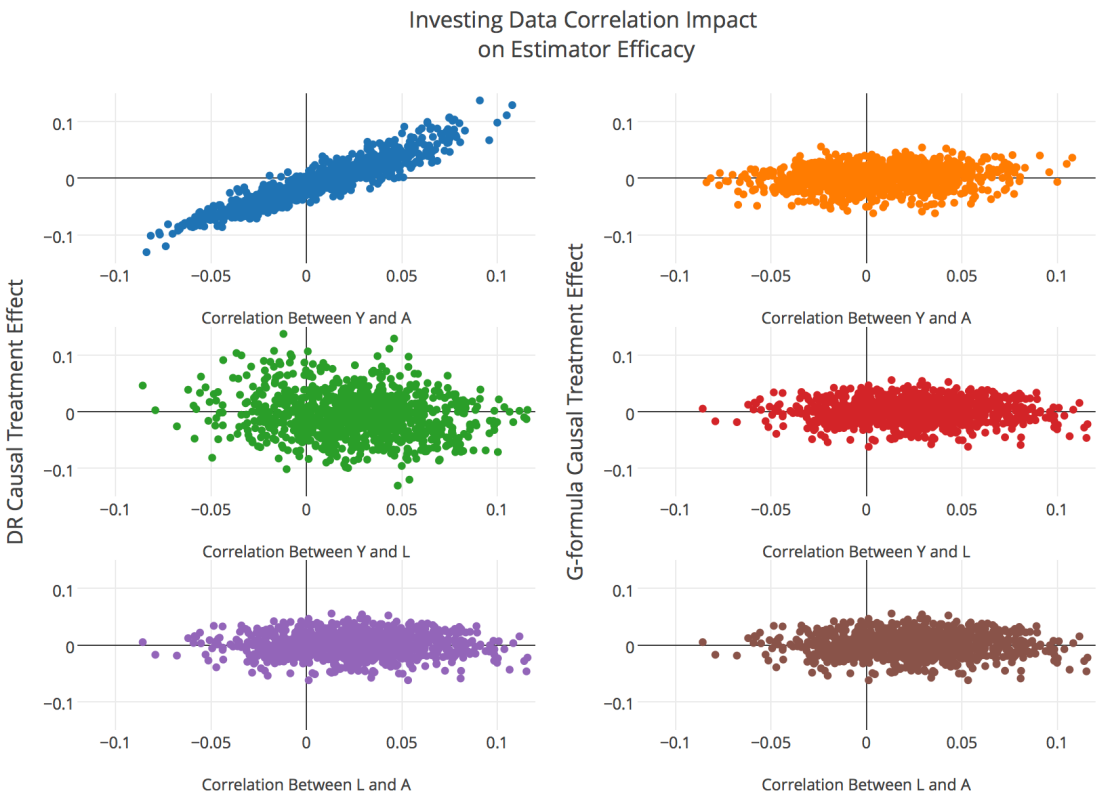

Figure 4.2: Various plots showing the relationship between underlying data correlations and estimated causal treatment effects using the two methods. Each data point is representative of one dataset. The same 1,000 datasets are used for each effect estimation.

4.3 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,

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

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

These models were chosen because they are quite random, and it would be unlikely based on the knowledge of how the data was created that these would be strong models for prediction. 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.

The results of this testing are shown in Table 4.3 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 4.3 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
$\hat{\alpha}$ misspecified, s_{m-1} correct	0.030	0.521	0.0042
Both models misspecified	0.133	0.005	0.0355

Table 4.3: 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.

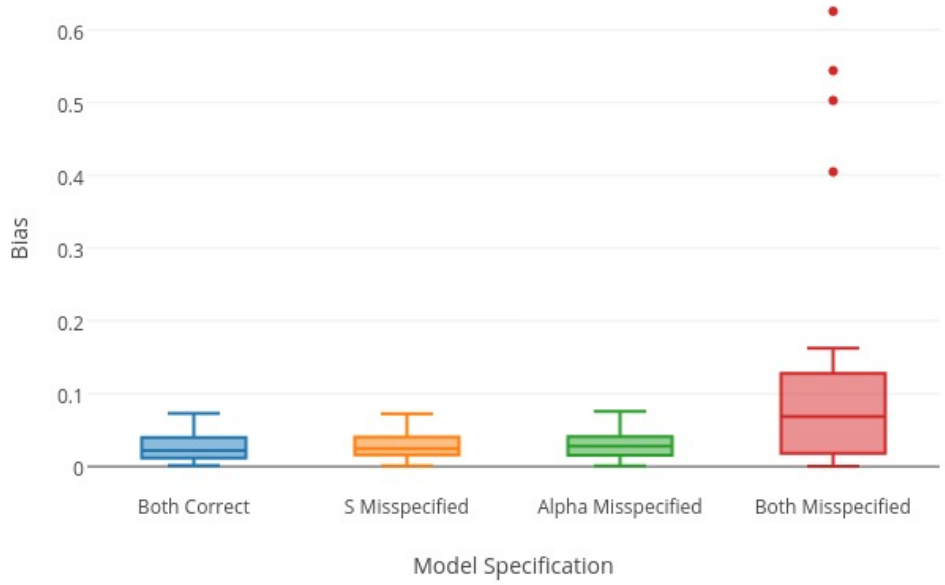

Figure 4.3: This boxplot shows the bias in the causal treatment effect estimates for varying combinations of models correctly specified. The fourth box, when both models are misspecified, shows the strongest bias and the most variance in that bias. However, the model remains robust when only one model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

4.4 Testing Multiple Robustness

With the evidence that the model was indeed doubly robust, it 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 wrong.

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

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

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

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

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 $s_3, \ldots, s_{j-1}, \pi_j, \ldots, \pi_K$ for $j \in \{4, \ldots, 12\}$, significant bias is introduced.[‡] However, no bias is introduced when the correctly specified models are of the form $\pi_3, \ldots, \pi_{j-1}, s_j, \ldots, s_K$ for $j \in \{4, \ldots, 12\}$. This implies that the doubly robust method is actually robust more 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 ??.

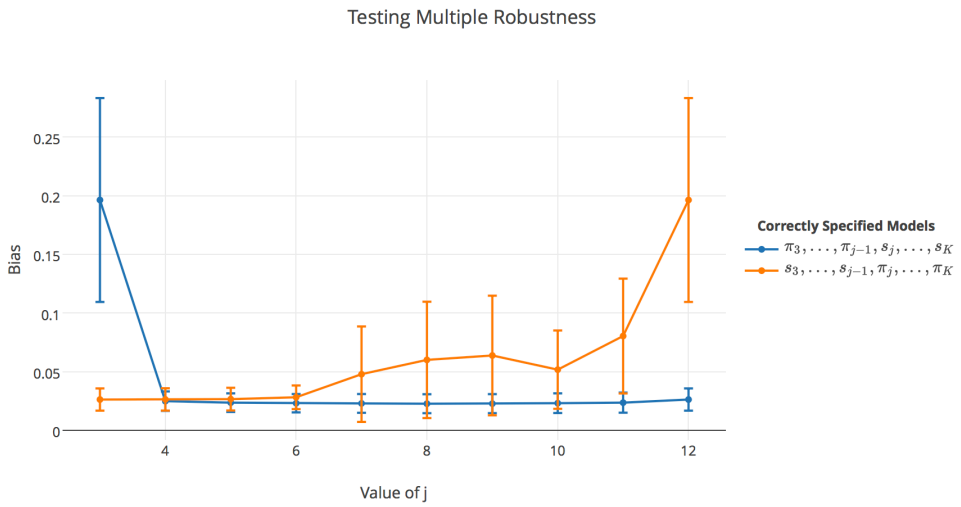

Figure 4.4: Plot showing the results of testing for multi-robustness. The models are correctly specified as given in the legend. Note that the end points of $j = 3$ and $j = 12$ correspond to the opposite end for the other line and represent the model where only one of the two models is correctly specified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

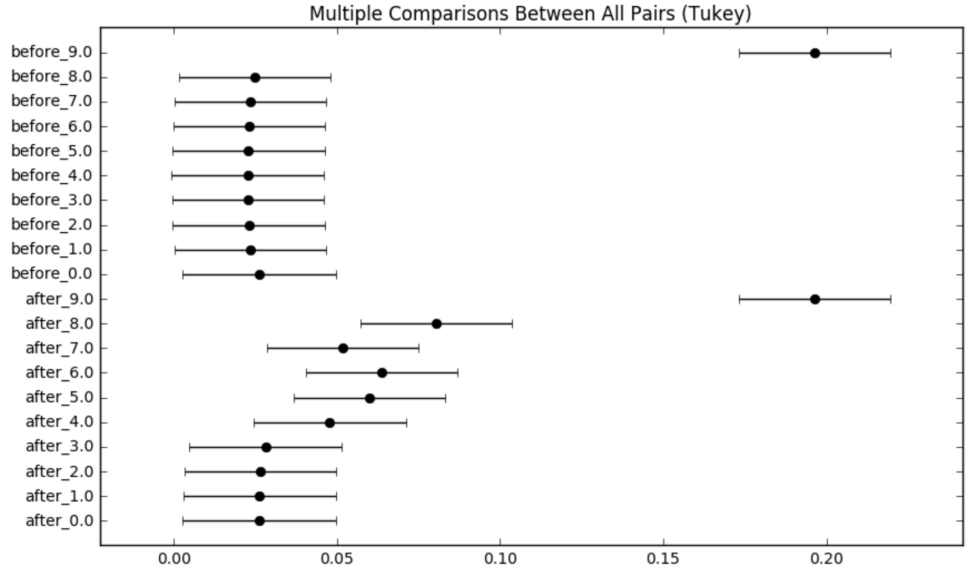

Figure 4.5: A plot of the confidence intervals of each of the model specifications in the testing of multiple robustness. The y-axis indicates which model, where the number is the value of j in the following models, the before model is $\pi_3, \ldots, \pi_{j-1}, s_j, \ldots, s_K$ and the after model is $s_3, \ldots, s_{j-1}, \pi_j, \ldots, \pi_K$. 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 viable methods of estimating causal treatment effect are presented here: g-formula and doubly robust estimation. The evidence presented here provokes 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, but it is more robust and much more efficient.

These methods have never been implemented in this form in Python before, a change which 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 restructured in order to be widely used and implemented. Capability must all be extended for including an extensive number of covariates.

The evidence regarding the robustness of the doubly robust method and the increased efficiency give this implementation the capacity to have quite an impact on many fields of science. In particular, massive datasets present as a cumbersome obstacle for the field of medical research, to which causal inference methods could be applied, possibly proving many novel discoveries. This could potentially save the field of medicine millions, even billions, of dollars in years to come.

A

Code

The following code are the major functions written to perform the investigation here. The functions follow the protocol lain out in Chapter 4.

FinalThesisFunctions

March 19, 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:
            x_Y = 0.5 + U
            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):
    columns = ["time", "gamma_0", "gamma_1", "gamma_2", "gamma_3", \
              "gamma_4", "gamma_5", "gamma_6"]
    train_columns = ["L1_1", "L1_2", "L1_3", "A_1", "A_2",  "A_3"]
    L1_model_df = pd.DataFrame(columns = columns)
```

```
    for ii in range(1, (max_time+1)):
        temp_df = df[df.time == ii]
        if ii == 1:
            L1_model = sm.Logit(np.asarray(temp_df["L1"]), \
                       np.asarray(sm.add_constant(temp_df[["L1_1", \
                       "A_1"]]))).fit();
            L1_model_df = L1_model_df.append(pd.DataFrame([ii] + \
                          [L1_model.params[i] for i in range(0,2)] \
                          + ["Nan"] + ["Nan"] + [L1_model.params[2]]\
                          + ["Nan"] + ["Nan"], index = columns)\
                          .transpose(), ignore_index=True)
        elif ii == 2:
            L1_model = sm.Logit(np.asarray(temp_df["L1"]), \
                       np.asarray(sm.add_constant(temp_df[["L1_1",\
                       "L1_2", "A_1", "A_2"]]))).fit();
            L1_model_df = L1_model_df.append(pd.DataFrame([ii] + \
                          [L1_model.params[i] for i in range(0,3)]\
                          + ["Nan"] + [L1_model.params[i] for i \
                          in range(3,5)] + ["Nan"], index = columns)\
                          .transpose(), ignore_index=True)
        else:
            L1_model = sm.Logit(np.asarray(temp_df["L1"]), \
                       np.asarray(sm.add_constant(temp_df[train_columns]\
                       ))).fit();
            L1_model_df = L1_model_df.append(pd.DataFrame([ii] +\
                          [L1_model.params[i] for i in range(0,7)],\
                          index = columns).transpose(), \
                          ignore_index=True)
    return(L1_model_df)


#########################################################################
##[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):
    columns = ["time", "zeta_0", "zeta_1", "zeta_2", "zeta_3", "zeta_4"]
    train_columns = ["L1", "L1_1", "A_1", "A_2"]
    A_model_df = pd.DataFrame(columns = columns)

    for ii in range(0, (max_time+1)):
        temp_df = df[df.time == ii]
        if ii == 0:
            A_model = sm.Logit(np.asarray(temp_df["A"]), np.asarray(\
                      sm.add_constant(temp_df[["L1"]]))).fit()
```

```
            A_model_df = A_model_df.append(pd.DataFrame([ii] + \
                        [A_model.params[i] for i in range(0,2)]\
                        + [float("nan")] + [float("nan")] +\
                        [float("nan")], index = columns).transpose(),\
                        ignore_index=True)
        elif ii == 1:
            A_model = sm.Logit(np.asarray(temp_df["A"]), np.asarray(\
                      sm.add_constant(temp_df[["L1", "L1_1", "A_1"]]\
                      ))).fit()
            A_model_df = A_model_df.append(pd.DataFrame([ii] + \
                        [A_model.params[i] for i in range(0,4)] +\
                        [float("nan")], index = columns).transpose(),\
                         ignore_index=True)
        else:
            A_model = sm.Logit(np.asarray(temp_df["A"]), np.asarray(\
                      sm.add_constant(temp_df[train_columns]))).fit()
            A_model_df = A_model_df.append(pd.DataFrame([ii] + \
                        [A_model.params[i] for i in range(0,5)],\
                        index = columns).transpose(),ignore_index=True)
    return(A_model_df)


#########################################################################
##[FUNCTION] simulation_run calculates the causal effect over an
## established number of repetitions 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)

    ### 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]))

    values[1] = np.sum(np.array([L1_model_df.ix[0,][i] for i in \
```

```
            [1,2,5]])*np.transpose(np.array([[1.0]*reps,\
            list(values[0]),[A_test[0]]*reps])), axis = 1)
p_v = np.exp(values[1])/(1+np.exp(values[1]))
values[1] = np.random.binomial(n=1, p = p_v)
prod = prod*p_v

values[2] = np.sum(np.array([L1_model_df.ix[1,][i] for i in \
            [1,2,3,5,6]])*np.transpose(np.array([[1.0]*reps, \
            list(values[1]),list(values[0]), [A_test[1]]*reps,\
            [A_test[0]]*reps])), axis = 1 )
p_v = (np.exp(values[2])/(1+np.exp(values[2])))
values[2] = np.random.binomial(n=1, p=p_v)
prod = prod*p_v



for jj in range(3, max_time+1):
    values[jj] = np.sum(np.array([L1_model_df.ix[jj-1,][i] \
                 for i in range(1,8)])*np.transpose(np.array(\
                 [[1.0]*reps,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])),\
                 axis = 1)
    p_v = (np.exp(values[jj])/(1+np.exp(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 = (np.exp(Y_exp)/(1+np.exp(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] 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)
```

B

Extra Math

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

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.