My project

A project by me

Mitchell Cameron

2024-07-31

doing a thing and hopefully done soon

Table of contents

Preface

```r
print("hello")
```

```
[1] "hello"
```

1 Introduction DONT READ THIS YET. ITS STILL BLAAA

1.1

1.2 Purpose and Intended Audience

1.3 Open Science

1.4 Layout of Book

This book is written using Quarto which is an open-source scientific and technical publishing system.

A background to causal inference and machine learning is provided in Chapter Chapter 2. This background is non-exhaustive but should provide an overview of all knowledge required to understand the methods and applications in this book. For readers with prior knowledge of causal inference, Chapter (**background?**) is unlikely to provide new information.

Following this background information I discuss #number of methods and provide applications leading to a strong understanding. These methods include Meta-Learners in Chapter (**metalearners?**), ...

1.5 Interpretability vs Accuracy

There exists a tradeoff between interpretability and accuracy across different machine learing and conventional statistical models. On one hand, models can be interpretable where reserachers can understand how a model

1.6 Machine Learning and Causal Inference

Given the lack of interpretability offered by many machine learning methods, there is often scepticism of causal learning. Cite Econometrics as a discipline has seldom interacted with machine learning as the interpretability of regression methods are favourable. While it is true that there are restrictions

The area where machine learning and causal inference collide is often relating to heterogeneous treatment effects. The literature on HTE's is diverse and rapidly growing. It seems that

make sure to emphasises the fundamental problem in terms of machine learning. why ml is a challnedge is because we dont observe the treatment effect so we cant rely on a hold out set to provde it works.

1.6.1 Two Competing Frameworks

1.7 Datasets and Languages

This book exclusively uses the R programming language as R provides an extensive list of packages for causal infernce and machine learning unavailable in other statistical programming languages. All data used in this book is publically available. The preparation and processing of all data is found in Section 7.1.

1.8 Reproducability

```r
set.seed(88)
```

1.9 Theming

```r
library(ggplot2)
library(showtext)
```

```
Loading required package: sysfonts
```

```
Loading required package: showtextdb
```

```r
font_add_google(name = "Source Sans Pro", family = "Source Sans Pro")

showtext_auto()

custom_ggplot_theme <- theme_classic(base_size = 11, base_family = "Source Sans Pro") +
    theme(
      text = element_text(color = "#333333"),
      plot.background = element_blank(),  # No plot background
      panel.background = element_blank(),  # No panel background
      axis.text = element_text(color = "#333333"),
      axis.title = element_text(color = "#333333", face = "bold"),
      legend.text = element_text(color = "#333333"),
      legend.title = element_text(color = "#333333", face = "bold",),
      plot.title = element_text(size = 14, face = "bold", color = "#333333"),
      plot.subtitle = element_text(size = 12, face = "italic", color = "#333333"),
      strip.text = element_text(face = "bold", family = "Source Sans Pro",
                                color = "#333333"),
      legend.position="bottom")

theme_set(custom_ggplot_theme)
```

2 Background: Causal Inference and Machine Learning

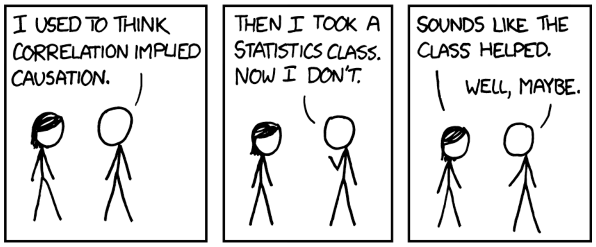

I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

Figure 2.1: This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. Source imgs.xkcd.com/comics/correlation.png

2.1 Causal Inference

2.1.1 What is Causal Inference?

Causal inference is a field of study concerned with determining cause-and-effect relationships between variables in observational and experimental data. Causal inference methods often utilise counterfactual reasoning to estimate the causal effect of an exposure, allowing researchers to make causal claims about the impact of interventions or treatments on outcomes. Ultimately, the goal of causal inference is to provide insights into the mechanisms driving observed phenomena, guide decision-making, and inform policy interventions aimed at improving outcomes in various domains such as healthcare, public health, economics, and social sciences. In this background chapter I discuss key ideas in causal inference such as the potential outcomes framework, common estimands, assumptions, graphical displays, and heterogeneous treatment effects.

2.1.2 Potential Outcomes Framework

The potential outcomes framework, also known as the Rubin Causal Model, was introduced by Rubin (1974) and builds upon the work of Splawa-Neyman (1923). The framework dominates how researchers think about causal inference and is highly influential in causal inference methodology. Rubin defines a causal effect as a defined comparison between two states of the world. The first state is where a treatment is applied resulting in a certain outcome. The second state is where the treatment is **not** applied resulting in a certain outcome. Hence we have two *potential outcomes*, one with the treatment and one without.

Consider a binary treatment variable, T_i, that takes the value of 1 if an individual i receives the treatment and 0 if they do not. Treatment for an individual T_i, is a random variable with the realisation t_i. We refer to the absence of treatment as the *control* state. Let $Y_i(1)$ and $Y_i(0)$ be the two potential outcomes for unit i under treatment $T_i = 1$ and control $T_i = 0$ respectively. For an individual, their causal effect, δ_i, is the individual treatment effect (ITE) and is simply the difference between these two potential outcomes:

$$\tau_i = Y_i(1) - Y_i(0) \tag{2.1}$$

Note 1: Fundamental Problem of Causal Inference

We are confronted by a clear problem because we only observe the outcome under either treatment or control. It is logically impossible for an individual to simultaneously both receive the treatment and not receive the treatment. For example, if I take medication to relieve a headache and my headache improves, I could never know what would have happened if I did not take the medication. This leads to the commonly discussed *fundamental problem of causal inference* - it is impossible to observe an individual's outcome under both potential outcomes. A counterfactual, the counter to the observed outcome, is infeasible and can never be practicably known.

It is important to differentiate between notation used to denote potential outcomes and observed outcomes. Different texts have used conflicting notation which I aim to make clear. The two potential outcomes for individual i are random variables denoted by $Y_i(1)$ and $Y_i(0)$ under the treated and control state respectively. The observed outcome for i is written as y_i^{obs} in general or $y_i^{obs}(1)$ and $y_i^{obs}(0)$ under treatment or control. Finally, we often attempt to predict outcomes for i which will be written as $\hat{y}_i$ in general which can also be under treatment or control as $\hat{y}_i(1)$ and $\hat{y}_i(0)$.

Much of the causal inference literature has involved to trying to find a counterfactual to compare outcomes between a treated a control individual. The fundamental problem requires clever experiemental or quasi-experiemental design and statistical methodology to resolve.

2.1.3 Estimands

In causal inference, there are multiple parameters of interest which we may wish to estimate. A desired or target parameter is called an estimand. The preferred estimand depends on the motivating example, discipline, and intended interpretation of results.

The most basic estimand is the *average treatment effect* or the ATE. This is the most general estimand as it is the average treatment effect on all individuals in the population regardless of whether they receive the treatment or not. This can be written as:

$$\begin{aligned} \text{ATE} &= E[\tau_{ATE}] \\ &= E[Y_i(1) - Y_i(0)] \\ &= E[Y_i(1)] - E[Y_i(0)]. \end{aligned} \tag{2.2}$$

A closed form solution for the estimate of the ATE is simply a difference in means between the group. Under certain conditions this can be an unbiased estimate of the ATE and is calculate as follows

$$\widehat{\text{ATE}} = \hat{\tau}_{ATE} = \frac{1}{N_t} \sum_{i=1}^{n} (y_i | t_i = 1) - \frac{1}{N_c} \sum_{i=1}^{n} (y_i | t_i = 0) \tag{2.3}$$

, where N_t and N_c are is the number of treated and control cases respectively.

The second parameter of interest is the *average treatment effect on the treated* or ATT and is the contrast between the potential outcomes of those that would actually receive the treatment. In other words, considering only those where $T_i = 1$, what is the treatment effect? This can be written as:

$$\begin{aligned} \text{ATT} &= E[\tau_i | T_i = 1] \\ &= E[Y_i(1) - Y_i(0) | T_i = 1] \\ &= E[[Y_i(1) | T_i = 1] - E[Y_i(0) | T_i = 1]. \end{aligned} \tag{2.4}$$

The final parameter that is commonly of interest is the *average treatment effect on the control.* This estimand is similar to the ATT but on the control observations. The ATC is the contrast between the two potential outcomes for individuals which are actually in the control. This is also known as the average treatment effect on the untreated or the ATU. It can be written as:

$$\begin{aligned} \text{ATC} &= E[\tau_i | T_i = 0] \\ &= E[Y_i(1) - Y_i(0) | T_i = 0] \\ &= E[[Y_i(1) | T_i = 0] - E[Y_i(0) | T_i = 0] \end{aligned} \tag{2.5}$$

For the ATT and ATC, no closed form solutions exist and so estimation is completed using G-methods to obtain contrasts of potential outcomes Naimi, Cole, and Kennedy (2017). These estimands fall outside the scope of this research but are important to know for those learning about causal inference.

2.1.4 Assumptions in Causal Inference

We must make some assumptions for the potential outcomes framework to be logically coherent and for estimands to be identifiable. Firstly, we must assume *independence*, implying the potential outcomes are independent of assignment into T. This assumption is also known as unconfoundedness as there is no confounding relationship between the treatment and potential outcomes. Mathematically we can state independence as

$$(Y_i(1), Y_i(0)) \perp\!\!\!\perp T_i. \tag{2.6}$$

This independence assumption implies exchangeability which means that we could switch the individuals in the treatment and control groups and obtain the same potential outcomes for each individual. An weaker assumption is conditional independence which states that assignment into treatment is random conditioned on some X. Hence,

$$(Y_i(1), Y_i(0)) \perp\!\!\!\perp T_i | X_i. \tag{2.7}$$

This requires that covariates must be known and measurable which may not always hold. Independence motivates the use of randomisation in experimental settings. Note that when using randomisation in a clinical setting, we are full independence which is a stronger assumption than conditional independence. Conditional independence implies exchangability between people with comparable propensity scores as if there was a randomised control trial.

A second assumption is *positivity*. This means that for each i, the probability of being in the treatment or control group is greater than zero and less than 1. Equally, $P(Y_i|T_i = 1) > 0$ and $1 - P(Y_i|T_i = 1) > 0$. If these condition is not met, it is theoretically impossible to obtain a potential outcome of the opposite state which is contradictory to the potential outcomes framework. If an individual is guaranteed to receive the treatment, it is logically impossible that they have a potential outcome in the control state and vice versa.

The third assumption is *consistency* between the potential outcome and observed outcome. For every i, the observed outcome under treatment equals the potential outcome under treatment and the observed outcome under control equals the potential outcome under control. Put mathematically, $Y_i^{obs} = Y_i(1)$ when treated and $Y_i^{obs} = Y_i(0)$ when under control . The relationship between the observed and potential outcomes leads to a switching equation which allows us to define the observed outcome, Y_i^{obs} as a function of the potential outcomes:

$$Y_i^{obs} = T_i Y_i(1) + (1 - T_i) Y_i(0) \tag{2.8}$$

Notice the logic of this equation, when $T_i = 1$ then $Y_i^{obs} = Y_i(1)$ as the second term zeroes out. Similarly, when $T_i = 0$ then $Y_i^{obs} = Y_i(0)$ as the first term zeroes out.

The final key assumption is called the *stable unit treatment value assumption* or SUTVA. This is a complex way of stating that there is no interference between units. More specifically, neither potential outcome $Y_i(0)$ or $Y_i(1)$ is affected by what treatment state any other individual received. To borrow terminology from economics, there are no 'externalities' or spillover effects from one individuals treatment status individual on another individual's potential outcomes.

Key Idea

In causal inference, especially when working with observational data, it is critical that these assumptions are considered. If these assumptions do not hold, any model, regardless of the modelling assumptions, will not have a causal interpretation. Unfortunately, there are no tests that can confirm if these causal assumptions hold and thus researchers must understand the context and data generating process in which they operate.

As these assumptions are so critical, I provide a critical review of causal inference assumptions in the context Zhou et al. (2021) who examine the impact of wildfires on excess Covid-19 cases and deaths.

2.1.5 Causal Relationships and Directed Acyclic Graphs

Given the very complex nature of many causal questions and relationships, it is important that we are able to communicate the underlying data generating proccess effectively. Directed acyclic graphs or DAGs are a graphical system for representing the direction and structure of relationships between variables in a causal context. They can be conceptualised as chains of causal effects strung together to display stuctural links in data. DAGs have some simple 'rules' which apply in all cases. By definition, effects are single-directional and do not contain loops -

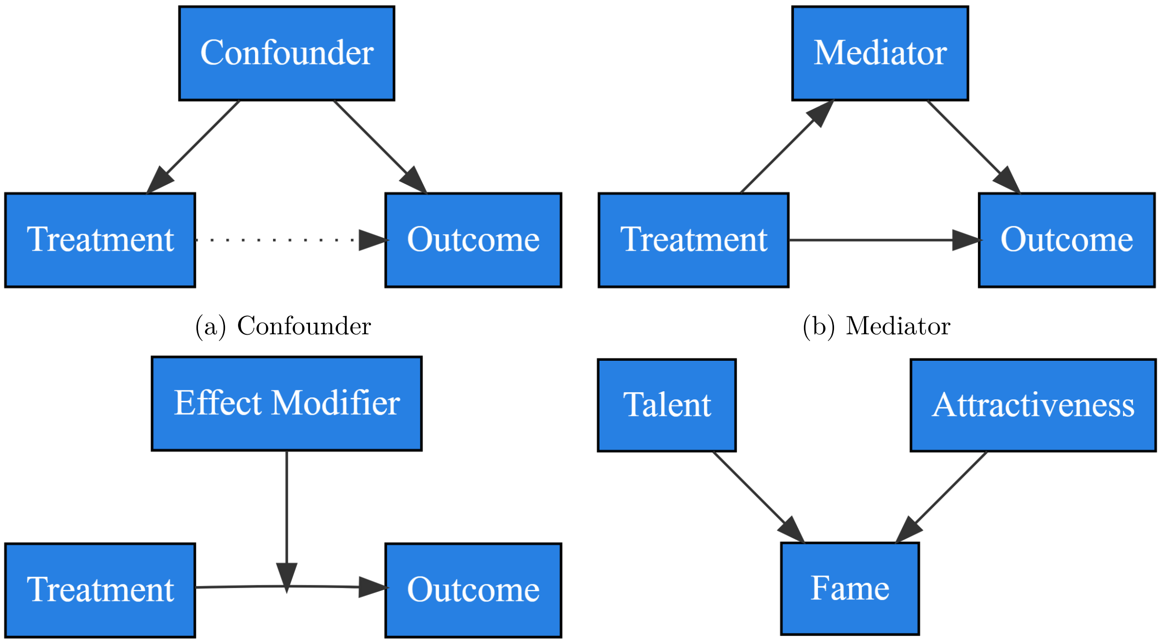
hence directed *acylical* graphs. To display bi-directional effects or reverse causality, researchers must use multiple DAGs. The reason for these two strict requirements is that reverse causality and bi-directional effects lead to significant over-complication of the causal process and result in infeasible problem sets. The three key relationships in the context of causal inference are confounding, mediation, and effect modification.

Confounding will be familiar to most readers as it is commonly discussed in introductory level statistics and quantitative courses. Confounding is where a variable has a causal effect on both assignment into treatment and the outcome variable. For example, consider a study investigating the relationship between physical activity (treatment) and improved cardiovascular health (outcome). The age of the individual will both determine how much physical activity an individual does and the cardiovascular health of that individual. If a confounding variable is not properly controlled for, we might attribute the effect of the confounder to the variable of interest, leading to incorrect conclusions about the causal relationship.

Mediation is a variable that mediates the effect between the treatment and observation. In our health example, physical activity might influence weight loss, and weight loss might subsequently affect cardiovascular health. Weight loss acts as a mediator between physical activity and cardiovascular health. Mediation analysis involves decomposing the total effect of an exposure on an outcome into direct and indirect effects through the mediator. It helps in understanding the mechanisms underlying the observed associations and identifying potential intervention targets.

Effect Modification, also known as interaction or effect heterogeneity, occurs when the relationship between an exposure and an outcome differ depending on the level of a third variable, known as the effect modifier. The effect of an exposure on an outcome varies across different levels of the effect modifier. Perhaps the effect of exercise on cardiovascular health is greater amongst older individuals than younger individuals. Effect modification is particularly relevant in this thesis and further discussion continues in later sections.

A more intuitive way to understand these relationships is with the visual such as in Figure 2.2. Figure 2.2a, Figure 2.2b, Figure 2.2c, and Figure 2.2d shows each of these relationships displayed as a DAG.



(a) Confounder

(b) Mediator

(c) Effect Modifier

(d) Collider

Figure 2.2: Directed Acyclical Diagrams

I do not elaborate on the DAGs further as there are many high-quality texts on the topic.[1] The key information in this subsection is the three relationships which can best be understood visually using DAGs.

[1] I particularly reccomend any interested reader use https://www.dagitty.net/ which allows users to draw DAGs and then convert them into LaTeX or R code for the corresponding R package. Additionally, Chapter 3 of Cunningham (2021a) provides and intuitive and application focused explanation of DAGs.

2.1.6 Experimental vs Observational Data

Working under the assumptions provided above, it is quite easy to determine the ATE which is a simple difference in means between the treatment and control groups. To overcome confounding effects, researchers can use experimental designs such as randomisation or a block design to theoretically guarantee unconfoundedness. In this case, the ATE can be calculated as the difference in means between treated and control group. In terms of the potential outcomes framework, under randomisation, the control group provides the counterfactual for the treatment group.

However a significant problem exists, it is impossible or unethical to use experiential methods for many causal questions. In many public policy applications, it is not possible to use a experimental design and randomisation. If a researcher wants to estimate the causal effect of raising the national minimum wage on unemployment, how would they use randomisation? In many medical applications, using an experimental design is highly unethical. For example, if a researcher wanted to determine the causal effect of drinking alcohol on brain development in teenagers, it would be highly unethical to provide alcohol to teenagers in a treatment group. If a criminological researcher wants to understand the causal impact of absentee fathers on a child's probability of committing a crime, it is impractical and unethical to force a child to be fatherless.

Given these issues, researchers must sometimes rely on observational data. This is data that is collected based on observations in the real world. When working with observational data, the assumptions specified above are often broken. In particular, there is often a confounding effect in data which is a clear break of the unconfoundedness assumption. Consider the causal effect of having a bachelors degree on lifetime earnings. IQ is a confounding variable which increases the chance of obtaining a bachelors degree but also increases lifetime earnings regardless of if that individual obtains a degree. If we contrast the mean lifetime earnings of degree-holders and non-degree holders, we can obtain an estimate of the ATE but it will be biased due to the confounding effect.

2.1.7 Treatment Effect Heterogeneity: Defined

The concept of the ATE is useful but *averages* can mask differing individual treatment effects inside a sample. For many treatments, the value of certain covariates, X, will influence how the treatment impacts the outcomes. There is treatment effect heterogeneity when the same treatment has different effects on different people. Perhaps a pain relief medication will relieve my headache after eating ice cream but cause a worse headache for another person.

The *conditional average treatment effect* or CATE describes the ATE conditioned on some quantity of a covariate. For example, individuals who are 40-50 years old and male. The CATE, also denoted as a function $\tau(x)$ is

$$\text{CATE} = \tau(x) = E[Y_i(1) - Y_i(0) | X_i = x] \tag{2.9}$$

Treatment effect heterogeneity is particularly of interest in medical contexts to study the impact of medications across different subpopulations. For example in medical studies, certain medications may be effective in men but have significant side effect when prescribed to woman. In marketing, certain advertisements may be more or less effective on an individual depending on their consumption patterns. If we could estimate the CATE then we could cater the delivery of the treatment to those who would benifit most or avoid treating those where the treatment is ineffective or harmful.

The CATE can be specified at different levels of granularity (how specific or general) depending on what is conditioned on. At its most granular the CATE is just the ITE when conditioned on covariates that only that individual has. On the other hand, the CATE can be more generalised and only condition on subgroup membership such as if someone is male or over 40 years old.

2.1.8 Interdisciplinary Applications of Causal Inference Methods

Causal questions exist in many disciplines. The tools used to make causal inferences are inter-disciplinary with different adaptations depending on research objectives and applications. Across disciplines, different phraseology exists, and it is important to be mindful of this when reviewing causal inference literature. A non-exhaustive list of disciplines, examples, and methodologies is provided for readers to grasp the wide variety of applications and the potential impact of causal research:

Economics, Public Policy, and Social Sciences: Practitioners and policymakers can use causal inference and determine causal effects of economic policies or social policies. Knowledge of causal effects can help with policy development and evidence-based decision making on policy issues. Card and Krueger (1994) use a differences-in-differences approach to study the impact of increasing the minimum wage on unemployment. Chu and Townsend (2019) use regression analysis and synthetic control methods to study the effect of medical marijuana legalization on crime rates.

Medicine, Genetics, and Epidemiology: Researchers can use causal inference to determine treatment effects of medications and guide the development of best-practice medical guidelines. Noroozi et al. (2020) use coarsened exact matching to study the relationship between methamphetamine use and HIV risk behavior among people who inject drugs. Tafforin and Segal (2022) use a co-twin control research design to student the causal effect of space travel on health and behavior.

Marketing, Management, and Human Resources: Businesses can use causal inference methods to assist in marketing, strategy, and organizational management leading to insights, growth, and profitability. Nabi et al. (2022) use augmented inverse probability weighting (AIPW) to estimate the effect of advertisement placement. Kovach (2018) uses propensity matching methods to examine the relationship between key performance indicators (KPI's) and attrition in corporate environments.

Agriculture and Horticulture: Agricultural researchers can use causal inference to improve efficiency, yield, profit, and sustainability by understanding the impact of farming processes or policies. Pan, Smith, and Sulaiman (2018) use regression discontinuity design to measure the causal effects of an agricultural advancement program on technology adoption and food security in Uganda. Key and McBride (2008) use instrumental variables to estimate how production contracts affects farm productivity in the United States.

Ecology, Climate Science, and Environmental Management: Researchers can answer important ecological questions and understand how certain interventions may impact on environmental health and outcomes. Ramachandra (2019) uses deep learning and satellite image time series to determine the treatment effects of various interventions on climate change. Andam et al. (2008) use matching methods to estimate the effectiveness of protected area networks in reducing deforestation in Costa Rica.

2.1.9 Linear Regression and Causality

Linear regression is well known method to the relationship between a dependent variable Y and one or more independent variables X. The relationship is assumed to be linear, and can be expressed by the following equation:

$$Y = \beta_0 + \beta_1 T_1 + \beta_2 X_1 + \epsilon \tag{2.10}$$

where Y is the dependent variable, β_0 is the intercept, β_1 is the coefficient on the treatment term, and β_2 is a confounder being controlled for. A *dummy* variable takes the value of 0 or 1 and can represent a binary treatment and β_1 is the estimate of the treatment effect. Such a simple linear model This means that β_1 estimates the average difference in Y between the group with $T = 1$ and the group with $T = 0$, controlling for other variables X.

Does Linear Regression Have a Causal Interpretation?

Generally linear regression does not have a causal interpretation - recall the adage *correlation does not imply causation*. However, if we assume that we know the direction of causality, there are no ommited or unobserved confounders, there is positivitiy with common support, the model is correctly specified, and that the treatment effect is homogeneous, then we could suggest a causal interpretation of the model in **?@eq-ols-model**. An example when this is the case would be in an RCT where the β_1 is the difference in means between the treatment and control groups.

2.2 Machine Learning

Machine learning or ML is a subset of artificial intelligence that involves the creation of algorithms that allow computers to learn from and make decisions or predictions based on data. In the realm of statistics in an ML age, models can be broadly classified into parametric and non-parametric. Parametric models make certain assumptions about the underlying data distribution and have a fixed number of parameters. Examples include linear regression and logistic regression. They are simpler and faster to use, but may not fit complex data well.

On the other hand, non-parametric models do not make strong assumptions about the data's underlying distribution. They can have an infinite number of parameters, allowing for more flexibility. Examples include decision trees and k-nearest neighbors. While they can fit complex data well, there drawbacks in terms of overfitting and loss of interpretability. Most ML methods are non-parametric including the methods discussed below.

2.2.1 Classification And Regression Trees

L. Breiman et al. (1984) introduces the Classification and Regression Tree, commonly known as CART, that partitions data according to a splitting criterion, resulting in an "if this, then that" interpretation. CART models are also widely known as a decision trees. The splits are recursive, meaning splits are applied upon previous splits, such as trees breaking into branches and then leaves. The splits are also *greedy* as each potential split only considers information available at that split instead of past or future splits. Each parent node is split to create two child nodes and the final nodes of a CART model are called terminal nodes.

For example, when classifying pets into cats versus dogs, the first split might be*"if barks"* and the second is *"heavier than 5 kg"*. The tree says *If it barks and is heavier than 5 kg, then it is a dog.*

A single classification tree typically uses the Gini index to determine its splits. Each split aims to maximize node purity, meaning the nodes contain the highest possible proportion of one class. The Gini index measures impurity, with lower gini values indicating higher purity. Intuitively, the aim of a classification tree's loss function is to minimize the misclassification rate of observations. By selecting splits that reduce the Gini index, the tree minimise classification errors and accuracy.

2.2.2 Ensemble Learning

Ensemble learning refers to a general framework of machine learning that combines together multiple simple models to create a better model. The philosophy behind ensemble learning is rooted in the wisdom of crowds principle, where the collective decision of multiple models often outperforms that of individual models.

Brieman (2001) introduces random forest which are an ensemble method that improves upon CART's predictive power and generalization ability. It builds multiple decision trees using bootstrapped samples of the dataset and selects random subsets of features for splitting at each node. The final prediction is then determined by aggregating the predictions of individual trees, typically through majority voting for classification or averaging for regression. A similar ensemble method to random forest is bagging or bootstrap aggregation. This ensemble uses bootstrapped samples of the data similar to random forests but without randomised feature selection at each split.

Friedman (2001) introduces the gradient boosting machine (GBM) that sequentially trains weak learners (e.g., shallow CART models) to focus on instances that were misclassified or poorly predicted by previous learners. Each subsequent model corrects the errors of its predecessors, gradually improving the overall model's performance. Boosting leverages the concept of ensemble learning, where combining multiple weak models leads to a stronger and more accurate predictor.

2.2.3 Cross Validations

Cross-validation is used to assess how a model generalizes to an independent dataset. It involves partitioning the data into subsets, training the model on some subsets (training set) and validating it on the remaining subsets (validation set). A common type is k-fold cross-validation, where the data is divided into k equal parts. The model is trained k times, each time using a different subset as the validation set and the remaining as training data. This technique helps in detecting overfitting, ensuring the model performs well on unseen data, and provides a more reliable measure of model performance.

2.2.4 Least Absolute Shrinkage and Selection Operator

Tibshirani (1996) introduces LASSO regularization, short for Least Absolute Shrinkage and Selection Operator, is a technique used in linear regression to prevent overfitting by penalizing the absolute size of the coefficients. It adds a penalty term to the ordinary least squares objective function, where the penalty is the sum of the absolute values of the coefficients multiplied by a regularization parameter lambda. Mathematically, it can be represented as

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j|$$

Lasso regularization encourages sparsity by driving some coefficients to exactly zero, effectively performing feature selection. This property is called shrinkage and makes lasso particularly useful when dealing with high-dimensional datasets with many irrelevant covariates.

Comparing with Ridge Regression, which uses the sum of squared coefficients as the penalty term, lasso tends to produce sparse models. Elastic Net, on the other hand, combines both L1 (lasso) and L2 (ridge) penalties, offering a balance between sparsity and stability. In the application of CRE's, the shrinkage property of lasso is of interest.

2.2.5 Moving Foward

Given that all of these conditions are hard to satisfy, there exists a vast range of tools practitioners can use to infer causality. In particular, machine learning provides new and powerful methods in causal inference which is the focus of this book.

3 Propensity Scores with Machine Learning

3.1 A Conventional Approach: Propensity Scores and Balance

In a randomised control trial (RCT), researchers believe treatment and control groups are similar because of randomisation. In this case, the similar groups are compatible and should not have systematic differences. For similar groups, the average treatment effect (ATT) is a contrast of means from Equation 2.3. In observational data, the exposure to a treatment is unlikely to be random, implying there may be systematic differences between groups. Systematic differences refer to consistent variations or disparities between groups in the study. These differences are not due to random chance but rather indicate a pattern or trend, perhaps due to selection-bias. As groups are not comparable, Equation 2.3 leads to a biased estimate of the treatment effect.

For example, consider the causal question: *"How much does obtaining a bachelors degree increase lifetime earnings?"*. Individuals who complete a bachelor's degree are not selected at random for university programs (treatment) and may have different observable attributes than those who do not attend a university (control). Perhaps those who attend university have higher academic abilities, higher motivation, or grew up with parents with higher income. Because of these systematic group covariate differences, a simple comparison of mean income could lead to attributing university attendance as the *cause* of higher incomes when the effect is confounded by the differences in covariates between groups. Recall that Figure 2.2a shows a confounding relationship. In this example, the confounding covariates are academic ability, motivation, and parental income that impact the probability of someone obtaining a bachelors degree. This discussion introduces the idea of *covariate balance* which is a key concept behind underlying propensity score methods.

> Note 2: What is Covariate Balance
>
> Covariate balance is the idea that covariates are approximately equivalent across treatment and control groups. If the distribution of each covariate are the same for each group, then those covariates are *balanced*. If covariates are similar across groups, then there should not be any confounding. Equally, similar covariates across groups implies exchangability between groups as the two groups should be similar (thus can be exchanged). There is a conceptual equivalence between covariate balance, unconfoundedness, and exchangeability meaning that Equation 2.6 is satisfied when covariates are balanced.

In bachelor's degree example, suppose that comparable treatment and control individuals are matched together to create balanced pairs. Between these pairs, covariates are balanced such as the same academic ability, motivation, parent income, geographic residence etc. Comparing the balanced matched pairs should result in a robust estimate of a bachelor's degree's impact on earnings because the individuals are exchangeable and satisfy Equation 2.7. The covariates are said to be "conditioned on" by matching individuals on these covariates. However, practically this matching is difficult to perform as exact matches cannot be made as the number of covariates increases. For example, finding two people with the same gender is simple but finding two people with the same gender, age, education, income, motivation, location, experience, and race is nearly impossible. Thus, there is a *dimensionality* problem as the dimension of the number of covariates increases.

Rosenbaum and Rubin (1983) offer a valuable tool for analysing observational data called the propensity score. The propensity score is the probability of treatment assignment conditioned on observed covariates. Essentially, the propensity score reduces the dimension of the number of covariates to a single dimension to avoid the dimensionality problem. Let the propensity score be denoted as $e(X)$ and be expressed as:

$$e(X) = P(T = 1|X). \tag{3.1}$$

A prediction of the probability of treatment based on the covariates is the best summary of each covariate. The covariate imbalance between bachelors degrees and controls arose from people self-selecting themselves into a bachelors degree programme because of these covariates. For example, people with higher motivation and academic ability are more likely to go to university. If it is the covariates that impact the probability of going to university, then a prediction of the probability of going to university based on these covariates should summarise the covariate effects.

Conditioning on this propensity score should balance the data and meet the conditional independence assumption stated in Equation 2.7. There are many sources that offer a comprehensive guide to propensity score methods such as (Cunningham 2021b, chap. 4) who provides applications and coded examples in R, Python, and Stata.

Note 3: Balance and Propensity Scores

Note that an RCT will satisfy Equation 2.6 as randomisation implies the potential outcomes are independent of the treatment assignment. Propensity score methods aim to satisfy Equation 2.7 as the potential outcomes are independent of the treatment status conditioned on some covariates.

The propensity score is conditioned on the covariates because the covariates are the predictors of the propensity score. Conditioning on the propensity score aims to replicate an RCT in observational data by balancing covariates between groups. When observations are conditioned on their propensity score, differences in outcomes can be confidently attributed to the treatment itself, rather than to pre-existing differences in covariates.

Two common methods that use propensity scores are propensity score matching (PSM) and inverse propensity weighting (IPW). PSM creates matched sets with similar propensity scores. IPW creates a balanced pseudo-population, where observations are weighted on the inverse of the propensity score. The pseudo-population is created by up-weighting observations with a low propensity score and down-weighting observations with a high propensity score.

At a high level, the conditioned property of the propensity score is translated into a model by using PSM or IPW. King and Nielsen (2019) provide a notable criticism of propensity score matching, which is a very interesting read. In the following examples, IPW is used due to theoretical advantages and ease of software implementation.

3.1.1 Propensity Score Modelling with Logistic Regression

A conventional propensity score model uses logistic regression to predict a probability between 0 and 1. Models may be specified to include interaction terms and polynomial terms so the model captures complex trends in the data. There are a range of approaches for specifying a propensity score model, but the process is a heuristically driven art rather than science. (Brookhart et al. 2006; Heinrich 2010). One suggestion is to include two-way interaction terms between covariates and squared terms and then remove terms which are not statistically significant. Notably, many researchers do not discuss the specification of propensity models in their papers. Austin (2008) reviews 47 papers that use propensity scores and find few perform adequate model selection, assess balance, or apply correct statistical tests.

It is important to note that the true propensity score is never observable. A propensity score that is close to the theoretically true probability is well calibrated. Poorly calibrated propensity scores may result in poor balance and biased estimation of the treatment effect. Propensity scores may be poorly calibrated as covariates may be omitted by error, poorly measured, or be unobservable. Logistic regression may not predict calibrated scores if the true relationship is non-linear or involves complex interactions between covariates. Another important note is that the propensity model itself does not have an informative *causal* interpretation. In logistic regression, the coefficients are the log-odds of the treatment assignment for each variable which is not informative of the desired estimand.

The first application of machine learning in causal inference was to predict propensity scores. Despite this, logistic regression still appears to be the most common model for predicting propensity scores.

3.2 Probability Machines: Probability Theory and Machine Learning

Probability prediction is not a typical machine learning task. Supervised machine learning usually focuses on classifying observations into groups, or predicting continuous outcomes. Probability prediction is a hybrid of these tasks, aiming to predict the continuous probability an observation belongs to a certain class.Machine learning methods that predict probabilities are sometimes called probability machines.

Probability machines are valuable in applications requiring calibrated probability predictions. For example, probability machines can predict loan defaults or other adverse events in finance. In marketing, they estimate the likelihood of customer response to a campaign. Gamblers and bettors want robust probability predictions to enhance their betting strategies. Probability machines can be applied wherever calibrated probability predictions are needed.

Probability machines offer many advantages over parametric methods like logistic regression:

1. **Improved Calibration**: Probability machines often provide better-calibrated predictions by capturing complex data relationships.

2. **Flexible Modelling**: Unlike parametric methods like logistic regression, probability machines don't rely on assumptions of additivity or linearity, allowing them to model intricate relationships that parametric models miss.

3. **Efficient Feature Selection**: These machines automatically select features, making them ideal for high-dimensional datasets where manual selection is impractical.

4. **Handling Missing Data**: Probability machines handle missing data robustly, minimizing the need for extensive data reprocessing and imputation.

5. **Simplified Data Exploration**: By exploring complex data structures in a data-driven way, probability machines simplify model specification. For instance, tree-based models remain unaffected by adding squared or interaction terms, streamlining the modeling process.

In causal inference, probability machines can predict better calibrated propensity scores and better estimate treatment effects. This discussion aims to clarify the use of probability machines in causal inference given the unique requirements of propensity score specification. Probability machines are theoretically complex and there are unanswered questions in this space.

Please note that this chapter assumes a reader is familiar with *CART (Classification and Regression Tree)*, *Boosting*, *Bagging (Bootstrap Aggregation)*, *Random Forests*, *LASSO (Least*

Absolute Shrinkage and Selection Operator), and *Logistic Regression.* These methods are briefly discussed in Section 2.2.

3.2.1 Choice of Loss Function and Probability Prediction

The loss function measures the difference between a model's predictions and the actual target values, serving as a measure of a model's performance. The best model exists at the minimum of the loss function. In standard least squares regression from Section 2.1.9, the loss function is the residual sum of squares stated as $\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. This loss function states the model must reduce the squared differences between the observed and predicted values. Different loss functions influence a model's behaviour so the choice of loss function is important.

Classification models predict the category that each observation belongs to not the probability of each class. For instance, in fraud detection, banks use classifiers to distinguish between fraudulent and routine transactions. Many classification loss functions minimize classification errors and improve accuracy as this results in the best classification. A loss function like the Gini index (introduced in Section 2.2.1) is effective for classification problems but it is unclear if this applies to probability problems. In other words, minimizing misclassification error may not lead to accurate probability predictions.

At a high level, to classify an observation, x_i as an A or B, a model needs to determine if $\Pr(x_i = A)$ is less than or greater than 0.5. If $\widehat{\Pr}(x_i = A) > 0.5$, then it is more likely to be an A and if $\widehat{\Pr}(x_i = A) < 0.5$ then it is more likely to be a B. Thus, if x_i is an A it is trivial if $\widehat{\Pr}(x_i = A)$ is 0.51 or 0.99 as this makes no difference to the classification as an A. But the difference between $\widehat{\Pr}(x_i = A) = 0.51$ and 0.99 is extreme for a probability machine. It is important to understand that classification models are optimized for classification accuracy rather than probability prediction. This distinction affects the performance of ensemble methods like random forests or bagging ensembles that use classification trees for probability prediction.

3.2.2 Bagging and Random Forest as Probability Machines

Across an entire bagging or random forest ensemble, class probabilities are determined through a *vote count* method. Within that ensemble, each tree makes a class prediction based on the majority class in a terminal node. For instance, if x_i lies in a terminal node where 80% of the observations are classified as an A, that *individual tree* will classify x_i as A. The ensemble's overall prediction for x_i is derived from the proportion of trees that classify x_i as A or B. Thus the ensemble *counts votes* for each class across the ensemble. Let T be the total number of trees and b_t be the t-th tree in the ensemble. Let $\mathbb{I}(b_t(x_i) = A)$ be the indicator function that returns 1 when b_t predicts that observation x_i belongs to class A. The probability of class A for observation x_i is calculated as:

$$\Pr(x_i = A) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}(b_t(x_i) = A). \tag{3.2}$$

Olson and Wyner (2018) notes a potential bias towards predictions of 0 or 1 when trees in a bagged ensemble or random forest are highly correlated. Using a vote count method, an ensemble will bias predicted probabilities towards $\hat{P}(x_i = A) \in \{0, 1\}$ when trees are highly correlated. Imagine that each tree in the ensemble is perfectly correlated implying that each tree will make the the same class prediction for each x_i. For such an ensemble, the predicted probabilities will will be exactly 0 or 1 using a vote count. Of course an ensemble of identical trees is unrealistic but the intuition still applies in the real world where ensembles may have some degree of correlation. The larger the correlation, the more the probability predictions will exhibit a *divergence bias* towards 0 and 1. Notably, divergence bias is not problematic in classification applications, as a larger number of trees correctly classifying the observation is encouraging.

If x_i has a known membership of A, and an unknown $\Pr_{\text{true}}(x_i = A) = 0.6$, the ensemble might classify x_i correctly 90% of the time leading to $\widehat{\Pr}(x_i = A) = 0.9$. As a probability machine, the ensemble has overestimated the probability by 0.3 even though a 90% classification accuracy is excellent. To predict $P_{\text{true}}(x_i = A) = 0.6$, an ensemble needs to incorrectly classify x_i in 40% of its trees. However, bagging ensembles and random forests are designed to maximize classification accuracy and there is no incentive for the model to intentionally achieve a specific misclassification rate that aligns with the true probability.

To reduce tree correlation, bagging ensembles use bootstrap aggregation and train each tree on a randomly selected bootstrapped sample of the data. Random forests further reduce tree correlation by considering a random number of variables at each split, referred to as *mtry*. When *mtry* is equal to to number of predictors, the model considers all variables at each split and the random forest is equal to a bagging ensemble. A lower *mtry* should reduce the correlation between trees and decrease divergence bias as the structure of the tree is modified by the selected variables at each split. However, a lower *mtry* also introduces other theoretical problems.

Consider the scenario where the binary outcome (treatment and control) of the ensemble is strongly related to a single predictor and weakly related to other noisy predictors. If *mtry* is low then each split may not consider the strong predictor and more commonly splits on weak or noisy predictors. Each predictor has a chance of $\frac{mtry}{\text{number of predictors}}$ of selection at each split implying a lower *mtry* decreases the chance of a splitting on the strong predictor. Splits on the weak or noisy predictors may not result in a meaningful increase in node purity and successive splits may result in impure terminal nodes that poorly predict the class of x_i in each tree. Such an ensemble may have highly unstable probability predictions.

Additionally, consider there is a class imbalance and the majority of observations are classified as A not B. The terminal nodes of each tree within an ensemble are more likely to contain the majority class. Consequently, there is a *majority class bias* as each tree in the ensemble is more likely to misclassifying an observation as an A because the terminal nodes have a higher proportion of A due to the higher proportion of A's in the data overall.

Note 4: Class Imbalance and Machine Learning

When there is a difference in the number of observations in each class, this is called class imbalance. It is important to note that majority class bias exists in conventional classification tasks with machine learning. Bagging ensembles and random forest are well known to be sensitive to class imbalance meaning that class predictions are biased towards the majority (see Bader-El-Den, Teitei, and Perry 2019).

However, the class imbalance problem is particularly notable when predicting probabilities. The probability prediction from a vote count method is very sensitive to a change in the votes from each tree. Suppose that balanced data results in 80/100 trees classifying B as B and imbalanced data (more A than B) reduces correct classifications of B to 60/100. This results in a 20% margin of error in probability estimates but the classification remains as B.

Individually, a low *mtry* can lead to unstable probability predictions and class imbalance can create bias towards the majority class. But probability machines are particularly effected when there is both a low *mtry and* class imbalance. Because successive noisy splits (relating to a low *mtry*) result in impure child nodes, the effects of majority class bias are exaggerated. Without the ability to separate the classes, the majority class will dominate terminal nodes. If the ensemble was able to split on informative covariates each time (*mtry* is higher), then it should still be able to create pure splits even when there is some class imbalance. In other words, if there is a small class imbalance, reducing *mtry* may reveal majority class bias not visible at higher *mtry*'s. Equally, if there is low *mtry*, then even a small class imbalance can lead to majority class bias.

To exemplify these theoretical points, the National Supported Work (NSW) programme is a commonly discussed dataset in causal inference. The data results from a randomized controlled trial with 445 total participants, 185 in the program group, and 260 in the control group, so the true probability of treatment for each individual can be calculated as $185/445 = 0.42$ or 42%. Further information about this data is found in Section 7.1.1.

Randomisation should ensure that the probability of treatment is independent of the predictors and so all predictors should be noisy or weak. Although Figure 3.2 and Table 3.1a suggest some covariates do have a greater impact on the probability of participating in the programme, which echoes research by Smith and Todd (2005) who suggests that self-selection bias is prevalent in the NSW data.

Figure 3.1 shows both divergence bias and majority class bias using `randomForest()` to fit both the random forest and bagging ensemble. Recall that a bagging ensemble is a random forest model when *mtry* is equal to the number of predictors and so specifying `mtry = 7` in the `randomForest()` function fits a bagging ensemble. Additionally, logistic regression using the `gbm()` function provides a meaningful comparison.

```r
library(randomForest)
set.seed(88)
nsw_formula <- as.formula(as.factor(treat) ~ age + educ + re75 +
                          black + hisp + degree + marr)

logit_preds <- glm(nsw_formula, data = nsw_data,
                   family = binomial())$fitted.values
```

```
rf_mtry1_preds <- predict(randomForest(nsw_formula,
                          mtry = 1, data = nsw_data),
                          newdata = nsw_data, type = "prob")[, 2]

bagging_model <- randomForest(nsw_formula, mtry = 7, importance = TRUE,
                              data = nsw_data)

bagged_preds <- predict(bagging_model, newdata = nsw_data, type = "prob")[, 2]

library(ggplot2)
plot_pmachines <- function(pscores, plot_subtitle) {
  ggplot(nsw_data, aes(x = pscores, fill = factor(treat))) +
    geom_density(alpha = 0.6, size = 0.6) +
    scale_fill_manual(values = c("#e5e5e5", "#2780e3"),
                      labels = c("Control", "Participant")) +
    labs(subtitle = plot_subtitle, x = "Propensity Scores", y = "Density",
         fill = "Group:") +
    scale_x_continuous(expand = expansion(0), limits = c(0,1)) +
    scale_y_continuous(expand = expansion(0), limits = c(0,10)) +
    custom_ggplot_theme
}

p1 <- plot_pmachines(logit_preds, "Logistic Regression") + xlab(NULL) +
  theme(legend.position = "none") +
  annotate(geom = "curve", x = 0.6, y = 5, xend = 0.42, yend = 0,
```

```r
           curvature = .3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 0.6, y = 5, label = "True Probability",
           hjust = "left", color = "#333333", size = 3,
           family = "Source Sans Pro")

p2 <- plot_pmachines(rf_mtry1_preds, "Random Forest (mtry = 1)") + xlab(NULL) +
  theme(legend.position = "none")
p3 <- plot_pmachines(bagged_preds, "Bagging (Bootstrap Aggregation)")

library(patchwork)
p1 / p2 / p3 + plot_annotation(
  title = "Density Plots of Propensity Scores for NSW Data")
```

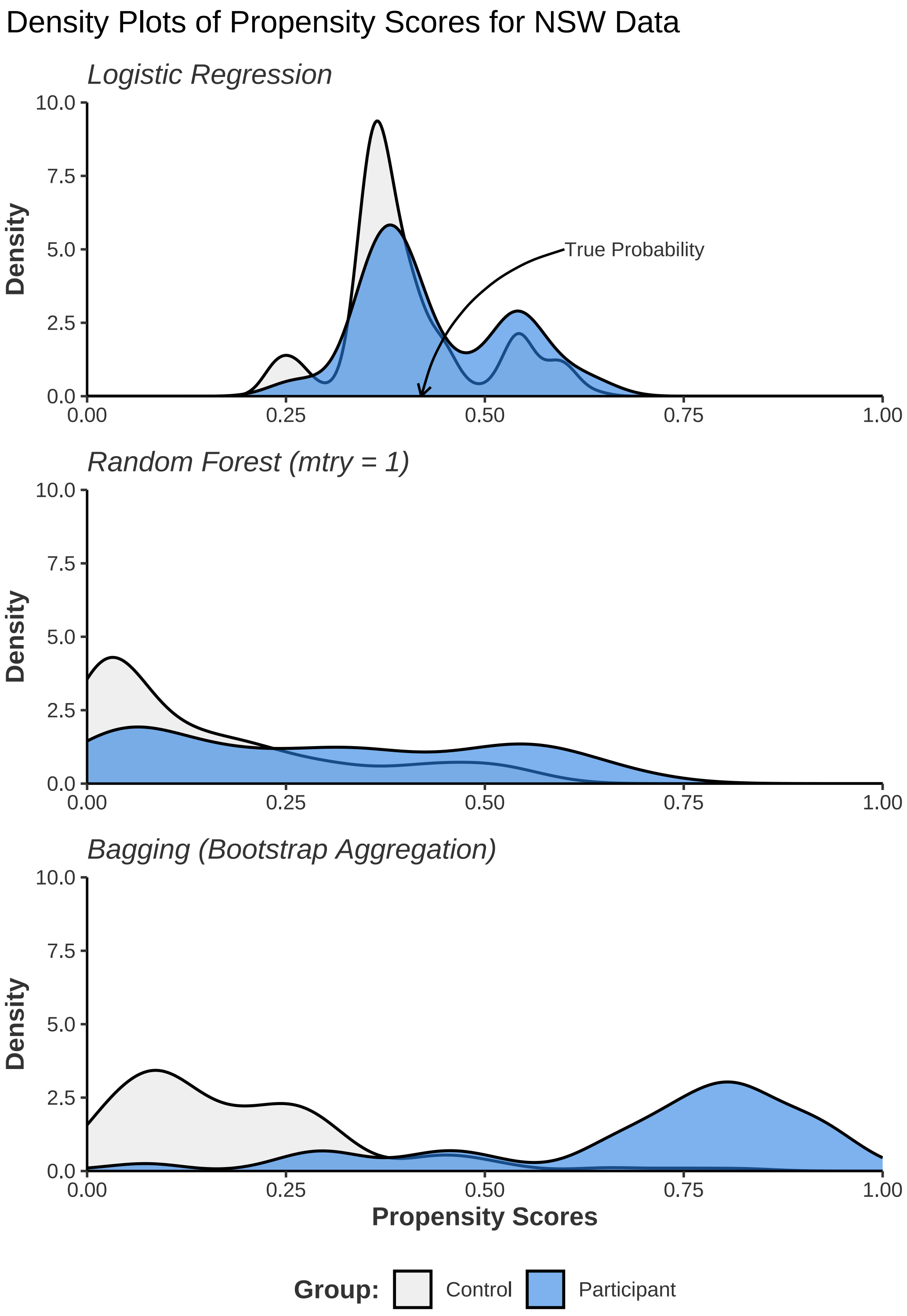


Figure 3.1: Shows kernel density estimates of propensity scores for each controls and participants in the National Supported Work programme. `randomForest()` fits a random forest with `mtry = 1` and bagging ensemble with `mtry = 7`. Uses the default values of `ntree = 500` and `nodesize = 1`.

```
library(ggplot2)
library(tidyverse)
imp <- as.data.frame(importance(bagging_model))
imp <- cbind(vars = rownames(imp), imp)
imp <- imp[order(imp$MeanDecreaseGini),]
imp$vars <- factor(imp$vars, levels = unique(imp$vars))

imp %>%
  pivot_longer(cols = matches("Mean")) %>%
  ggplot(aes(y = vars, x = value, fill = name)) +
  geom_bar(stat = "identity", width = 0.8, show.legend = TRUE,
           position = position_dodge(width = 0.8), color = "black", size = 0.6) +
  facet_grid(~ factor(name, levels = c("MeanDecreaseGini", "MeanDecreaseAccuracy")), scales =
  scale_fill_manual(values = c("#e5e5e5", "#2780e3")) +
  scale_x_continuous(expand = expansion(c(0, 0.04))) +
  labs(
    title = "Variable Importance",
    x = "% Decrease if Variable is Omitted from Model",
    y = "Variable Name"
  ) + custom_ggplot_theme +
  theme(
    legend.position = "none"
  )
```

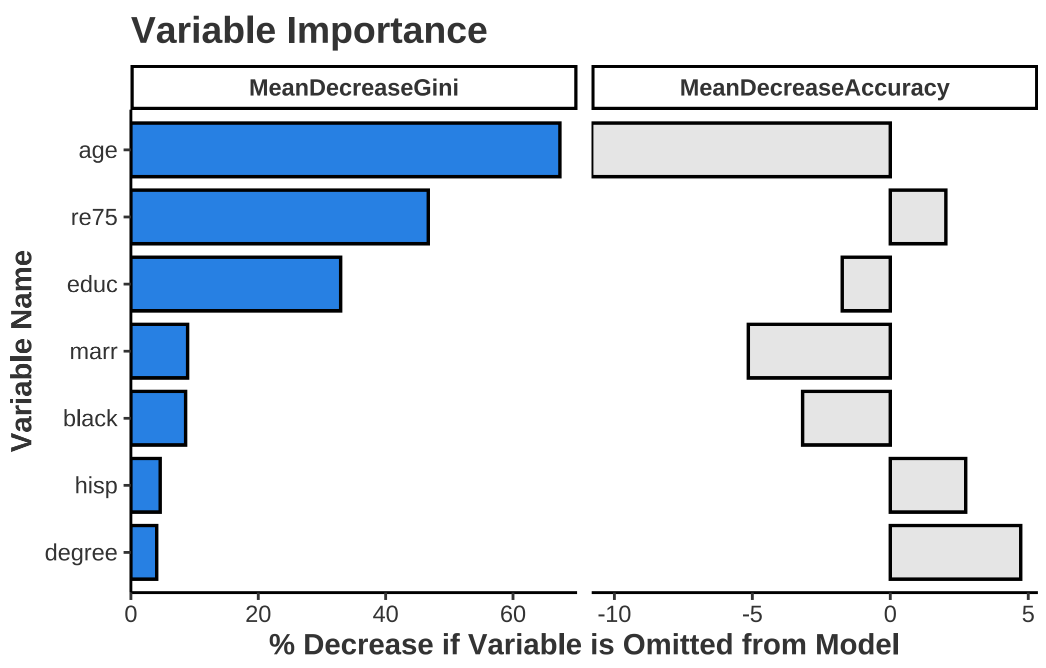


Figure 3.2: The figure compares the variable importance assigned to each variable from a baggin ensemble. The data originates from the National Supported Work programme. The difference in relative important of some variables indicates that randomisation may not have created exchangability between the groups.

Figure 3.1 shows the logistic regression model has identified a central tendency and most propensities are between 0.25 and 0.75 which roughly aligns with the true probability. The bagging ensemble has clear evidence of divergence and the majority of predictions are outside 0.25 and 0.75 which is likely related to tree correlation. For the random forest with $mtry = 1$, a significant number of the treatment and control observations are centred near the control area $(T = 0)$ with a wide range of other predictions. Recall that the control group is the majority class. Reducing $mtry$ from 7 to 1 reveals the majority class bias reinforcing the theoretical discussion that a combination of low $mtry$ and class imbalance is especially troubling. The models over predicts the majority class and has unstable predictions otherwise. Both random forest and bagging ensembles have performed poorly compared to the true probability of 0.42%.

The tuning of *mtry* faces double jeopardy and is another important area of discussion in probability machines. The selection of *mtry* is typically completed in with a classification loss function such as accuracy or out-of-bag error. Olson and Wyner (2018) compares tuning *mtry* measured by classification accuracy and mean square error of known simulation probabilities and finds that the optimal value of *mtry* for classification differs from probability prediction.[1] In other words, if a grid search finds that $mtry = 3$ is optimal for a classification task, this does not imply that $mtry = 3$ is optimal for predicting probabilities so the tuning of *mtry* is difficult.

Random forests and bagging ensembles seem to be troubled as probability machines but this does not mean that bagging and random forest cannot perform well. In various simulation studies, they perform excellently as discussed in Section 3.2.5. Perhaps the nature of the data is informative for the potential success of a random forest or bagging ensemble.

Heuristically, divergence bias and majority class bias will most effect a probability machine when there is considerable overlap between groups. If there is overlap and a central region of true probabilities, then the effects of divergence bias may be very pronounced. Similarly, common overlap may make it even harder to increase purity in child nodes, as the covariates will lack clear split points. When combined with weak predictors relating to a low *mtry*, the terminal nodes of each tree may be relatively impure leading to a majority class bias. Alternatively, if true probabilities exist near 0 or 1 and there is a clear separation of class, divergence bias may trivially effect probability estimation as the probabilities already exist in that region. If there is a clear separation of class, then weak predictors relating to a low *mtry* may still create meaningful splits and pure terminal nodes. It is worth noting that propensity score methods require datasets with overlap to meet the assumptions required to determine causality.

[1]Note that tuning *mtry* for the mean square of probability prediction is only possible by design of the simulation study and is not possible in applications, as the true probability is unknown.

3.2.3 Gradient Boosting Machines as Probability Machines

Moving beyond classification trees in random forests or bagging ensembles, Friedman (2001) introduced the *Gradient Boosting Machine* (GBM). A GBM sequentially constructs CART trees to correct errors made by previous trees. Employing a gradient descent process, each new tree is fit on the pseudo-residuals of the previous iteration. This means that with each iteration, the GBM takes a gradient step down the global loss function, incrementally minimizing the loss function to reach a minimum. The update rule for the model after each iteration can be expressed as:

$$\hat{p}_i^{(t)} = \hat{p}_i^{(t-1)} + \lambda \cdot b_t(x_i),$$

where λ is the learning rate, and $b_t(x_i)$ is the b-th regression tree fitted on the pseudo-residuals of the previous regression tree. In words, the current overall iteration t, is a combination of the previous model plus the current iteration scaled by a learning rate.

A learning rate controls the contribution of each weak learner to the final model. By using a small learning rate, the machine learns slowly so that it can slowly descend the loss function. This allows for finer adjustments during the iterative process to better capture patterns in the data.

GBMs can be generalized to many different applications by minimizing a different loss function which can be specified as any continuously differentiable function. For binary outcomes, a GBM employs multiple regression trees and a logistic function to transform regression predictions into probabilities. Specifically, the logistic function used is:

$$\hat{p}_i = \frac{1}{1 + \exp(-\text{model}(x_i))}.$$

This logistic function is the same as in logistic regression, so a GBM with a binary class is sometimes called boosted logistic regression. The ensemble aims to minimize the Bernoulli deviance, which is equivalent to maximizing the Bernoulli log-likelihood with logistic regression. Maximizing the log-likelihood ensures that the predicted probability distribution is as close as possible to the true probability distribution given the data. The full GBM model, $f_T(x)$ after T iterations can be written as:

$$f_T(x_i) = b_1(x_i) + \lambda \sum_{t=1}^{T} b_t(x_i).$$

Inside a base tree, each split considers all variables and makes the most informative split to descend the loss function using gradient descent. GBMs utilize many weak learners, such as a regression tree with a single split called a regression stump. However, additional splits enable the model to capture interactions between terms, which may increase probability calibration in complex or high-dimensional datasets.

By outputting probability predictions and avoiding the flaws of vote methods in other ensemble techniques, as well as allowing a probability distribution-based loss function optimal for probability prediction, GBMs stand out as a highly effective probability machine. Since GBMs predict probabilities from a logistic function, they avoid problems associated with a vote count method. Additionally, there are no difficult parameters to tune, such as *mtry* in a random forest. The implementation and workflow to fit a GBM for propensity scores is discussed in Section 3.3.1.

3.2.4 Overfitting

Overfitting is a common concern when fitting machine learning models, as models can capture noise and random variations in the training data. An overfit model will typically show excellent performance on the training data but will perform poorly on new, unseen data because it cannot generalise beyond the specific patterns of the training set. For instance, consider a machine learning algorithm used by a bank for fraud detection. In this scenario, an overfit model would struggle to classify transactions correctly as it has learned the noise and specific variation in the training data rather than the underlying patterns of fraud. Cross validation or test/train splitting can prevent overfitting to ensure a model can generalise to unseen data.

However, the model is not required to generalise a propensity score model as different datasets will have a different model. Instead, the emphasis of predicting propensity scores is to create balance in the data. A model is effective if it balances covariates between groups, even if it is overfit in a conventional sense.

Note 5: Overfitting in Logistic Regression

There is limited research on how overfitting a logistic regression model affects estimating treatment effects. In logistic regression, overfitting occurs when there are too many parameters and so the maximisation of the log-likelihood function is difficult because of noise. One study that investigates overfitting in this context is Schuster, Lowe, and Platt (2016), who suggest a general rule that the number of observations per parameter should be between 10 and 20. When overfitting occurs, the variance of the estimated treatment effect increases because noise amplifies the magnitude of the coefficients, resulting in a small bias towards 0 or 1 because of properties of the logit function. Specifically, when using (non-augmented) propensity score weighting, the estimate of the treatment effect will have high variance as propensity scores close to 0 or 1 receive artificially inflated weighting.

Lee, Lessler, and Stuart (2010) simulates a comparison of machine learning methods for propensity score prediction and finds that an overfit CART model performs better than a pruned CART model in terms of balance and treatment effect estimation bias. While not conclusive, this suggests that conventionally overfit trees are appropriate and potentially beneficial for propensity score modelling.

If overfitting was to occur, this could be interpreted as balance between groups getting worse decreases with a higher model complexity. Although various software packages use a stopping rule to prevent this. As conventional advice states, creating balance should be the aim of estimating propensity scores.

3.2.5 Comparison of Machine Learning Algorithms: Simulation Results

A small body of simulation studies benchmarks probability machines for predicting propensity scores (see McCaffrey, Ridgeway, and Morral 2004; Setoguchi et al. 2008; Lee, Lessler, and Stuart 2010; Cannas and Arpino 2019; Tu 2019; Goller et al. 2020; Ferri-García and Del Mar Rueda 2020). Although these studies tackle the same problem, differences in simulation design and model implementation lead to a diverse range of perspectives on this issue. This variety reflects the complexity of the propensity score prediction.

Tu (2019) compares logistic regression, boosting, bagging, and random forests across different sample sizes, conditions of linearity and additivity, and treatment effect strengths. Boosting achieves the lowest bias ATE estimate in most scenarios and the lowest mean square error in all scenarios. Bagging ensembles and random forests perform poorly in both ATE estimate bias and MSE. The author notes that poor performance in bagging ensembles is likely due to correlated trees in the ensemble, leading to divergence bias. Random forests perform significantly better than bagging but both methods performed worse than boosting or logistic regression.

Despite poor theoretical properties as a probability machine, Lee, Lessler, and Stuart (2010) find that bagging results in the lowest standard error across many datasets.[2] This result is not surprising given that the bagging ensembles are trained on bootstrapped datasets, leading to lower variance and standard error. Although, this advantage is not likely of practical interest given that the small performance gain in standard error is at the expense of a considerable increase of bias.

Additionally, Lee, Lessler, and Stuart (2010) finds that logistic regression performs well in simple data structures with comparable bias to boosting and random forest, but with larger standard errors. In complex data structures, boosting shows low bias and outperforms logistic regression while maintaining low standard errors. Consequently, the study concludes that boosted CART achieves the best 95% coverage in all simulation scenarios, with 98.6% coverage.[3]

Cannas and Arpino (2019) also undergo a simulation study to assess machine learning methods for propensity score prediction. They compare logistic regression, CART, bagging ensembles, random forest, boosting, neural networks, and naive bayes and find that random forest, neural networks, and logistic regression perform the best. Notably, the simulation design only performs hyperparameter tuning for CART, random forest, and neural networks but not either of their boosting implementation. [4] This is a weakness of their study design and thus their

[2]In this case, the standard error is the dispersion of the standardised mean difference (effect size) across 1000 simulated datasets.

[3]In this context, the coverage is the proportion of times that the true treatment effect is within the 95% confidence interval across the number of simulations. This author implements 1000 simulations of each scenario.

[4]Cannas and Arpino (2019) provide a replication package for their simulation study online and their hyperparameter tuning is process transparent. The authors fit two GBMs using the `twang` and `gbm` package in R. The hyperparameter values provided to these untuned boosting models are contrary to heuristics and may

findings may be more informative of the relative performance of tuned versus untuned models. Although, the finding that random forest performs well when tuned is significant.

Goller et al. (2020) adds diversity to the simulation study literature by exploring an economics context, experimenting with imbalances between treated and control observations, and incorporating LASSO and probit models.[5] Probit regression achieves the best covariate balance, with LASSO also performing well. In contrast, the random forest model performs poorly, showing imbalance statistics with several orders of magnitude higher than those of probit or LASSO. To perform feature selection, a probit model with many interactions and polynomial terms is specified, and a LASSO penalty shrinks covariate coefficients to zero. Probit regression stands out for its superior covariate balance, while LASSO also delivers satisfactory results. The random forest model underperforms with significantly higher imbalance statistics compared to probit and LASSO.

Based on a review of the literature, the findings can be distilled into five important points:

1. Probability machines can predict propensity scores with excellent performance and their implementation should be considered in most scenarios. Although, a logistic regression approach may be preferred because of simplicity while still providing adequate performance in simple data structures.

2. In cases of non-linearity or non-additivity in the data, probability machines often achieve better covariate balance and lower bias of treatment effect estimates than logistic regression. This is significant as propensity scores are frequently used in observational studies with complex data structures (Rosenbaum and Rubin 1983).

lead boosting to perform poorly regardless of theoretical benefits discussed in Section 3.2.3.

[5]Goller et al. (2020) calculates the bias of the treatment effect using the average of the estimates from logistic regression, random forest, and LASSO models as the *true* treatment effect. Thus, the covariate balance table offers a clearer view of each method's performance.

3. Bagging ensembles perform poorly, a finding replicated across multiple studies.

4. Random forests can perform excellently when hyperparameters are satisfactorily tuned.

5. Further research should consider parametric methods with LASSO, Ridge, or Elastic Net penalties to assist in feature selection. Simulation study evidence for predicting propensity scores is limited despite attractive properties of these methods.

6. A tuned GBM stands out with strong theoretical support, excellent simulation performance, and superior software implementation and documentation. Specifically, this GBM will use the Bernoulli deviance as a loss function due to theoretical benefits. Implementations of GBMs such as AdaBoost.M1 have no simulation study evidence.

7. A good practical approach seems to be a trial-and-error approach of fitting multiple model specifications, then considering covariate balance for each model.

3.3 Implimentation and Hyperparameter Tuning with `WeightIt` and`gbm` in R

Based on Friedman (2001), the `gbm` package implements a *Generalized Boosting Machine.* Here, the "generalized" is because the package provides generalisations of the boosting framework to other distributions such as Bernoulli, Poisson, and Cox-proportional hazards partial likelihood of class probability predictions. Although this implementation very closely follows Friedman (2001) who introduced the gradient boosting machine. `gbm` also supports stochastic gradient boosting, which performs random bootstrap sampling for each tree using the `bag.fraction` parameter.

To fit and tune a GBM for propensity scores, wrapper packages facilitate optimal hyperparameter tuning for covariate balance. An effective approach involves fitting the model and computing balance statistics at each hyperparameter combination. Since the `gbm` package does not support this type of tuning, a wrapper package like `WeightIt` is necessary. `WeightIt` allows for hyperparameter tuning based on covariate balance and inverse propensity weighting (IPW). `WeightIt` supports hyperparameter turning of `shrinkage`, `interaction.depth`, and `n.trees`. Once the best model is identified, propensity scores are predicted inside `WeightIt`. These can be used inside `WeightIt` to perform IPW or extracted for other implementations. `WeightIt` also supports an offset meaning that logistic regression predictions are supplied to the `GBM` package.

Multiple sources, including package documentation and other research, suggest values for hyperparameters (see McCaffrey, Ridgeway, and Morral 2004; Ridgeway et al. 2024). A very low learning rate, such as 0.01 or 0.0005, allows a smooth descent of the loss function. The model should include a high number of trees, with $10,000$ or $20,000$ being a typical default value. While this may seem excessive, it is required when a low learning rate is used. A grid search process should consider many options including a very high number of trees and even though the optimal model may contain fewer trees. While GBMs often use shallow trees like stumps, allowing a few splits per tree can better model non-linearity and additivity. The package default allows for 3 splits. Based on anecdotal experience, 1 to 5 splits per tree is optimal, consistent with recommendations by McCaffrey, Ridgeway, and Morral (2004).

Another package, `twang`, proves functionality to tune the number of trees, but there are no inbuilt options for tuning of other hyperparameters and so accessory packages such as `caret` must be used. Although `twang` has other useful functionalities which users may wish to implement.

3.3.1 Hyperparameter Tuning and Workflow

The `WeigthtIt` package seems to have the best options for hyperparameter tuning and integration with a package for assessing balance called `cobalt`. The best information for this package can be found on this website or accessed with `vignette("WeightIt")` inside R after installation using `install.packages("WeightIt")`.

A workflow for hyperparameter tuning in `WeightIt` may be completed as follows:

1. Specify the `criterion` option, which specifies the measure of the *"best model"*. The available options are the options that the `cobalt` can compute. A simple option to choose may be the average standardised mean difference (SMD) across all covariates called `sdm.mean` or the smallest maximum SDM across covariates called `sdm.max`.

2. Set the number of trees high. The package default is `n.trees = 10000` for binary treatments, but this may be too small depending on the learning rate. Typically, it is best to increase the number of trees to allow slow learners to reach their minimum criterion. There is no modelling downside to a larger number of trees other than computation time as the model will predict propensity scores with a smaller `n.tree` if optimal.

3. Specify the grid search for the depth of the tree called `interaction.depth` and the learning rate called `shrinkage`. These values can be specified using `c()` such as `shrinkage = c(0.0005, 0.001, 0.05, 0.1, 0.2, 0.3)` or as integers such as `interaction.depth = 1:5`. These particular values are heuristically selected *suggestions* of good starting values. Additionally, an offset can be considered by performing a grid search across `offset=c(TRUE,FALSE)`.

4. The model is fit and a grid search is performed. The tune grid and balance statistics can be retrieved with `my_weightit_object$info$best.tune`.

5. The best model should be inspected and to determine if the initial grid is appropriate. If the selection of the best model is at the boundary of a grid search, then a new grid should be created and step 3 and 4 are repeated. For example, if the initial fit is completed with `interaction.depth = 1:5` and the best fit is 5, then a new search can consider `interaction.depth = 3:7` so that the local area around 5 can be searched.

6. Experiment with `bag.fraction`, which means each tree will consider a drawn proportion of observations equal to `bag.fraction`. Iteratively changing `bag.fraction` and assessing balance at each value should be practical. Consider 0.5, 0.67, and 1.

7. Assess balance of covariates and model fit. Covariate balance can be assessed with a balance table or visualisation of the variables using `love.plot()` such as Figure 3.4.

8. The tuning process is stated and reported. Balance tables are presented and discussed. Comparison to other methods of estimation if relevant.

9. Estimation and reporting of treatment effect.

3.4 Example: NSW Jobs Dataset Using R

For demonstration, propensity scores are estimated following the workflow discussed in Section 3.3.1 to estimate inverse propensity weights (IPW). The NSW jobs dataset arises from a randomised setting as described in Section 7.1.1. Randomisation should eliminate structural differences between groups, but Rosenbaum and Rubin (1983) notes that randomisation only addresses structural balance and does not account for chance imbalance. To address this, propensity scores can mitigate any remaining chance imbalance, providing a more accurate estimate of the treatment effect. This example will include the fitting process of a GBM using `WeightIt` and a logistic regression model using `glm()`. Additionally, balance statistics will be computed leading to a robust estimate of the treatment effect. All code to replicate this process and results is provided.

> **Note 6: Inverse Probability of Treatment Weighting**
>
> Inverse probability of treatment weighting or inverse propensity weighting (IPW) adjusts for confounding in observational data by weighting individuals based on the inverse of their probability of receiving the treatment they actually got. This method creates a *pseudo-population* where treatment assignment is independent of observed covariates, similar to a randomized controlled trial. In this re-weighted population, the treatment and control groups should be have covariate balance, allowing for unbiased estimation of treatment effects. Essentially, IPW simulates random treatment assignment by rebalancing the sample, thereby eliminating confounding and enabling more accurate causal inferences.

3.4.1 Step 1-6: Model Fitting and Tuning

The `glm()` function will fit a conventional propensity score model with logistic regression in R. Logistic regression is performed by specifying the family to be the `binomial()`. Recall the `nsw_formula` is specified in Section 3.2.2

```
nsw_logit_pmodel <- glm(nsw_formula, data = nsw_data,
                        family=binomial())                        ①


nsw_logit_pscores <- nsw_logit_pmodel$fitted.values               ②
```

① Fits a logistic regression model using the `glm()` function specified to be a logistic model with `family=binomial()` using the previously created `nsw_formula`.

(2) Extracts the fitted values (propensity scores) from the model.

Using the propensity score column of `nsw_data`, the `WeightIt` package will perform IPW and assign a weight to each observation such that the pseudo-population should exhibit covariate balance. The model object will be called `nsw_logit_weight`.

```r
library(WeightIt)
nsw_logit_weight <- weightit(nsw_formula, data = nsw_data,           (1)
                             ps = nsw_logit_pscores,                  (2)
                             estimand = "ATE")                        (3)
```

(1) Specifies the formula and data.

(2) Provides `weightit()` with the propensity scores from the logistic regression function. Note that in practice this can be completed within the `weightit()` function with `method = "glm"`. The separate estimation of the propensity scores is for illustrative purposes.

(3) Specifies the estimand as the average treatment effect or ATE. For the purposes of demonstration, this is an arbitrary choice.

A GBM model for propensity scores can be specified using `method = "gbm"` inside the `weightit()` function. To ensure consistent results, running `set.seed(88)` will ensure each tree uses the same `seed` if `bag.fraction` less than 1. The model is fit using the heuristically suggested starting values. Note that this model may take approximately 30 second to fit as a grid search procedure is computationally intensive. Additionally, the best tuning specification is printed to assess if the initial tuning grid is appropriate.

```r
set.seed(88)
nsw_boosted_weight <- weightit(nsw_formula, data = nsw_data,         (1)
                               method = "gbm",                        (2)
```

```
                                       estimand = "ATE",
                                       shrinkage = c(0.0005, 0.001, 0.05, 0.1, 0.2, 0.3), ③
                                       interaction.depth = 1:5,
                                       bag.fraction = 1,                                  ④
                                       offset = c(TRUE, FALSE),
                                       criterion = "smd.mean",                            ⑤
                                       n.trees = 10000)
print(nsw_boosted_weight$info$best.tune)                                                 ⑥
```

① Specifies the formula and data.

② Specifies the propensity score prediction method to be a GBM and the estimand to the ATE.

③ Performs a grid search over these values of the learning rate and depth of tree.

④ Requires the model to use every observation in every tree, meaning the model will not perform stochastic gradient boosting. The function will will fit an offset and level GBM and select the specification with the best balance.

⑤ Defines the optimisation criteria to be the tune with the lowest average standardised mean difference (SMD). Additionally, the number of trees will be 10000 which is the package default.

⑥ Prints the tune details of the model with the best covariate balance.

```
  shrinkage interaction.depth distribution use.offset best.smd.mean best.tree
6       0.3                 1    bernoulli      FALSE    0.02253485      2392
```

The best balance across all tuning combinations yields an average SMD of 0.023 showing strong balance. Note averages can conceal extremes and a low average SMD does not mean all variables are balanced. A full balance table is presented in Section 3.4.2 accompanying a discussion of balance.

The best machine has a learning rate of 0.3 and contains 2392 decision stumps (trees with a depth of 1). The learning rate is on the boundary of the initial tuning grid showing that the tuning grid should be re-specified to include values near to 0.3. A reduction in the depth of tree and number of trees will reduce computation time.

The new tune grid will consider `shrinkage = c(0.25, 0.3, 0.35, 0.4, 0.45, 0.5)` as this allows the GBM to consider values between 0.2 and 0.3 and above 0.3 which were missing in the previous grid.

```r
set.seed(88)
nsw_boosted_weight2 <- weightit(nsw_formula, data = nsw_data,
                               method="gbm",
                               estimand = "ATE",
                               shrinkage= c(0.25, 0.3, 0.35, 0.4, 0.45, 0.5),
                               interaction.depth = 1:3,
                               bag.fraction = 1,
                               offset = c(TRUE, FALSE),
                               criterion = "smd.mean",
                               n.trees = 5000)


print(nsw_boosted_weight2$info$best.tune)
```

```
   shrinkage interaction.depth distribution use.offset best.smd.mean best.tree
11      0.45                 2    bernoulli      FALSE    0.01965492        95
```

Comparing the two iterations, there is a reduction from 0.022 to 0.02. The optimal tuning values are towards the centre of the tuning grid, implying that an adequate search of the local area has been completed. The best machine has a learning rate of 0.45, a tree depth of 2, and 95 trees. The learning rate is higher than expected, but this also explains why fewer trees are optimal.

Plotting the relationship between the number of trees and the average SMD is informative for the behaviour of the machine. Additionally, Figure 3.3 shows the optimal number of trees is highly variable. If the learning rate is set to `shrinkage = 0.05`, then the best balance is not achieved until near to $20,000$ trees.

```
low_shrinkage <- weightit(nsw_formula, data = nsw_data,
                               method = "gbm",
                               estimand = "ATE",
                               shrinkage = 0.05,
                               interaction.depth = 1,
                               offset = c(TRUE, FALSE),
                               criterion = "smd.mean",
                               n.trees = 40000)


library(ggplot2)
optimal_boost_plot <- ggplot(nsw_boosted_weight2$info$tree.val, aes(x = tree, y = smd.mean))
  geom_line(size = 1, color = "#2780e3") +
  labs(subtitle = "Optimal Tune",
       x = "Number of Iterations",
       y = "Average Standardised Mean Difference") +
  custom_ggplot_theme +
  xlim(0,500)


lowshrinkage_boost_plot <- ggplot(low_shrinkage$info$tree.val, aes(x = tree, y = smd.mean)) 
  geom_line(size = 1, color = "#2780e3") +
  labs(subtitle = "Low Learning Rate (shrinkage = 0.05)",
       x = "Number of Iterations",
       y = NULL) +
  custom_ggplot_theme +
  annotate(geom = "curve", x = 30000, y = 0.05,
           xend = low_shrinkage$info$best.tree, yend = 0.0231,
           curvature = 0.3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 31000, y = 0.05, label = "Minimum",
           hjust = "left", color = "#333333", size = 3, family = "Source Sans Pro")


optimal_boost_plot + lowshrinkage_boost_plot + plot_annotation(
  title = 'Number of Tree Iterations and Balance')
```

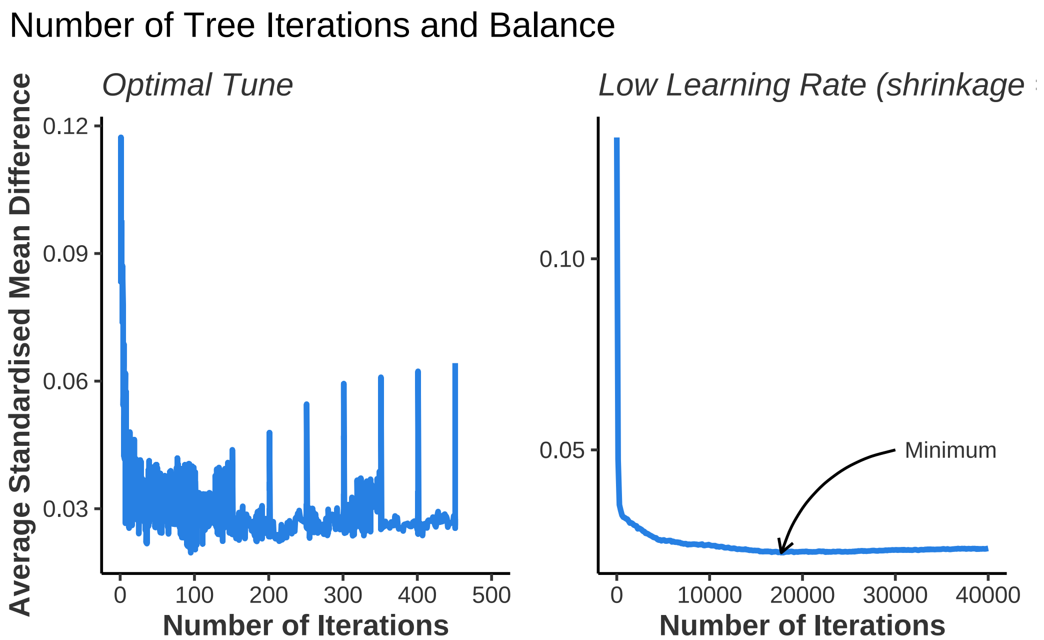


Figure 3.3: Average Standardised Mean Differernce (Covaraite Balance) and the number of interations. Please note the difference in horozontal scale between the two plots.

For the optimal machine fit, finding that balance worsens as the number of trees increases is just as informative as knowing the correct number of trees. Provided sufficient computational performance, a wide grid search is beneficial in the long run to ensure that each model specification reaches the best balance possible.

3.4.2 Step 7 and 8: Assessing Balance

The Importance of Discussing Balance

Assessing balance is crucial because it ensures that the treated and control groups are comparable on observed covariates. This comparability is essential for reducing confounding and making valid causal inferences. Without proper balance, differences in outcomes between the groups could be due to pre-existing differences rather than the treatment itself. Balance assessment helps to verify that the propensity score model has effectively adjusted for covariates, creating a pseudo-randomized scenario. This step is vital for the reliability and validity of the study's conclusions. King and Nielsen (2019) notes that many papers that implement propensity score methods do not assess or report a balance in their studies, which can undermine the credibility of the research process and make it hard for readers to understand why results are robust.

A good resource of information for assessing balance is documentation from the `cobalt` package, which can be viewed by running `vignette("cobalt", package = "cobalt")` in R.

`cobalt` is a powerful package to create tables and visualisations of to assess balance. The package also provides very good integration with other related packages such as `WeightIt` for IPW and `MatchIt` for propensity score matching. Balance tables are created using `bal.tab()`.

```
library(cobalt)                                                    ①
nsw_logit_btab <- bal.tab(nsw_logit_weight,                        ②
                          data = nsw_data,
                          stats = c("mean.diffs","variance.ratios"),  ③
                          binary = "std", continuous = "std",
                          thresholds = c(mean.diffs = 0.1))        ④
```

```
nsw_logit_btab <- nsw_logit_btab$Balance[-1,-c(2,3)]                          ⑤
```

① Loads the `cobalt` package. This assumes the package is already installed with `install.packages("cobalt")`

② Uses the `bal.tab()` fucntion to create balance statistics for the previously created `nsw_logit_weight` model.

③ Specifies the calculation of standardised mean differences and variance ratios for each covariate. The mean differences will be standardised for binary and continuous variables.

④ Sets a threshold of balance to be 0.1 to determine if a covariate is balanced.

⑤ Extracts the balance table of the `nsw_logit_btab` object and removes excessive columns. This is only completed for ease of visualisation and is not typically required.

Additionally, `bal.tab()` will create balance tables for the GBM method's IPWs and the raw data. For presentation, `dplyr` combines each of the individual balance tables for presentation using `kable` and `kableExtra`.

```
nsw_boosted_btab <- bal.tab(nsw_boosted_weight,
                            data = nsw_data,
                            stats = c("mean.diffs","variance.ratios"),
                            binary = "std", continuous = "std",
                            thresholds = c(mean.diffs = 0.1))


nsw_raw_btab <- bal.tab(nsw_formula,
                        data = nsw_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
                        thresholds = c(mean.diffs = 0.1),
                        s.d.denom = "treated")

# Extracts the balance table and removes unwanted columns.
nsw_boosted_btab <- nsw_boosted_btab$Balance[-1,-c(2,3)]
nsw_raw_btab <- nsw_raw_btab$Balance[-c(5,6)]
```

Table 3.1a shows that both logistic regression and the GBM have reduced imbalance. The raw data exhibits imbalance across age, years of education, if someone is gispanic, and if someone has a bachelors degree. Imbalanced datasets leads to biased treatment effect estimation so the estimate of the treatment effect in the raw data may be biased. In this example, logistic regression appears to achieve the best covariate balance although GBM achieves slightly better variance ratios.

3.4.3 Step 9: Results

Finally, the treatment effect can be estimated using `lm_weightit()` from the `WeightIt` package and `avg_comparisons()` from the `marginaleffects` package. `lm_weightit()` fits a linear model with a covariance matrix that accounts for the estimation of weights using IPW. Additionally, `avg_comparisons()` computes the contrast between the treatment and control group to obtain an estimate of the treatment effect.

These steps perform G-computation, meaning that potential outcomes are estimated under treatment and control for each observation (Naimi, Cole, and Kennedy 2017). The contrast of the mean of each of the two potential outcomes is the estimate of the treatment effect. Note that the outcome variable is `re78` which is real income in 1978 meaning that the income is

Table 3.1: Balance Table for NSW Data

```r
library(dplyr)
library(kableExtra)

collabels <- c("Type", "SMD", "Balanced", "Variance Ratio","Method")

rowlabels <- c("Age", "Education", "Income 1975","Black",
               "Hispanic", "Degree", "Married")

nsw_raw_btab$method <- "Raw Data"
nsw_logit_btab$method <- "IPTW: Logistic Regression"
nsw_boosted_btab$method <- "IPTW: Boosting"

combined_btab <- bind_rows(setNames(nsw_raw_btab,collabels),
                           setNames(nsw_logit_btab,collabels),
                           setNames(nsw_boosted_btab,collabels))

combined_btab$Variable <- rep(rowlabels,3)

combined_btab <- combined_btab[c(6,1,2,3,4,5)]

rownames(combined_btab) <- NULL

combined_btab$Balanced <- ifelse(
          combined_btab$Balanced == "Not Balanced, >0.1", "No", "Yes")

kbl(combined_btab[-6], digits=2,booktabs= T,align = "c",
      font_size=10) %>%
  kable_styling(full_width = T) %>%
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  column_spec(2:5, bold = F, width="3cm") %>%
  pack_rows(index = rev(table(combined_btab$Method)))
```

(a) Placeholder

Variable	**Type**	**SMD**	**Balanced**	**Variance Ratio**
Raw Data				
Age	Contin.	0.11	No	1.03
Education	Contin.	0.13	No	1.55
Income 1975	Contin.	0.08	Yes	1.08
Black	Binary	0.04	Yes	NA
Hispanic	Binary	-0.20	No	NA
Degree	Binary	0.28	No	NA
Married	Binary	670.09	Yes	NA
IPTW: Logistic Regression				
Age	Contin.	0.00	Yes	0.98
Education	Contin.	0.00	Yes	1.27
Income 1975	Contin.	0.01	Yes	0.80
Black	Binary	0.00	Yes	NA
Hispanic	Binary	0.00	Yes	NA

adjusted for inflation. Previously, the treatment indicator was the outcome variable because the propensity scores are a prediction of the treatment indicator.

```r
nsw_boosted_lm <- lm_weightit(re78 ~ treat * (age + educ + re75 + black +   ①
                              hisp + degree + marr), data = nsw_data,
                              weights = nsw_boosted_weight$weights)         ②


library(marginaleffects)                                                    ③
nsw_boosted_result <- avg_comparisons(nsw_boosted_lm, variables = "treat")
```

① Uses `lm_weightit()` to compute pseudo-outcomes. The formula here specifies an interaction between the treatment and all other variables. Note that `*` indicates multiplication in R.

② Specifies the `weights` from the `nsw_boosted_weight` object created earlier by the `weightit()` function. Intuitively, this is performing linear regression using the pseudo-population, where the pseudo-population is created weighting the data by `nsw_boosted_weight$weights`.

③ Computes a comparison between the potential outcomes as well as standard errors for inference.

Additionally, this process is followed for the logistic regression propensity scores and the results are combined in to a table for comparison.

```r
nsw_logit_lm <- lm_weightit(re78~treat*(age + educ +
                             re75 + black + hisp +
                             degree + marr), data = nsw_data,
                             weights = nsw_logit_weight$weights)
```

Table 3.2: Comparison of ATE Estimates

	ATE.Estimate	SE	P.Value	Lower.CI	Upper.CI
Logistic Regression	1610.79	668.49	0.02	300.58	2921.00
Generalized Boosting Machine	1609.95	669.42	0.02	297.91	2921.99

```r
nsw_logit_result <- avg_comparisons(nsw_logit_lm, variables = "treat")


nsw_comparisons_tab <- rbind(extract_comparison_results(nsw_logit_result),
                             extract_comparison_results(nsw_boosted_result))
rownames(nsw_comparisons_tab) <- c("Logistic Regression", "GBM")


kbl(nsw_comparisons_tab, digits=2,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
```

Table 3.2 shows that both estimates of the treatment effect are nearly identical at $1610 with logistic regression inferring a $0.86 larger treatment effect. Additionally, these results are statistically significant at the 5% level with nearly identical standard errors.

3.5 Replication Study

(**coffecite?**) aims to estimate the impact of the certification of coffee cooperatives on small-scale Ethiopian farmers' livelihoods. Certification is seen as a potential tool for socioeconomic change and environmental sustainability and so it is important to understand the impact on small-scale farmers. Propensity scores are used to balance covariates between certified and non-certified farmers, isolating the certification's effect on income.The paper did not assess the balance of propensity scores and it is difficult to replicate the results in the paper using best practice. However, this provides a good opportunity to assess covariate balance in the initial paper and the repeat the analysis using a machine learning propensity model.

3.5.1 Replication of Original Results

(**coffeecite?**) provides a replication package including Stata code that uses the psmatch2 function. Nearest neighbour matching with replacement and common support trimming is performed. Common support trimming means that any observations outside the commonly overlapping are are discarded. The results of the paper can be fully replicated using the `MatchIt` package inside R.

```
coffee_formula <- as.formula(certified ~ age_hh +
                  agesq + nonfarmincome_access + depratio +
                  logtotal_land + badweat + edu + gender +
                  years_cofeproduction + access_credit)


library(MatchIt)
library(marginaleffects)
coffee_rep_pmodel <- matchit(coffee_formula, data=coffee_data, distance="glm",
                             method="nearest", replace = T, estimand="ATT",
                             discard="both")


coffee_logit_md <- match.data(coffee_rep_pmodel)


coffee_rep_fit<- lm(percapitaincome_day_maleeq ~ certified, data = coffee_logit_md, weights=w
```

Table 3.3: Placeholder

	Estimate	SE	P.Value	Lower.CI	Upper.CI
Replicated Result	-0.15	0.74	0.84	-1.6	1.29

```r
replicated_result <- avg_comparisons(coffee_rep_fit, variables = "certified",
                vcov = TRUE,
                newdata = subset(coffee_logit_md, certified == 1),
                wts = "weights")
```

```r
replicated_result_tbl <- extract_comparison_results(replicated_result)
rownames(replicated_result_tbl) <- "Replicated Result"
kbl(replicated_result_tbl, digits=2,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
```

Table 3.3 shows the replicated result obtained by (**coffeecite?**). The intriguing finding of the paper is that the average treatment effect on the treated (ATT) of being certified on income is negative. That is, if a farmer becomes certified, this is predicted to decrease by $0.15 per day. Intuition and proponents of certifications schemes suggest that certification leads to an increase of income. If certification negatively impacted well-being in this way, it would call into question a significant effort to engage in certification and fair trade practices.

(**coffeecite?**) does not perform any discussion or consideration of balance in their paper and so it is not clear if their propensity score matching process has resulted in balanced covariates. A balance table created by the `cobalt` package will provide the required information for balance assessment which will be aided by a graphical visualisation using `love.plot()`.

```
library(cobalt)

coffee_rep_btab <- bal.tab(coffee_rep_pmodel,
                        data = coffee_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
                        thresholds = c(mean.diffs = 0.1),
                        s.d.denom = "treated")

coffee_rep_btab_ss <- coffee_rep_btab$Observations

coffee_rep_btab <- coffee_rep_btab$Balance[-1,-c(2,3)]

rowlabels <- c(
  "Household Age", "Squared Household Age", "Non-farm Income Access",
  "Log Total Land", "Dependency Ratio", "Bad Weather",
  "Education Level", "Gender", "Years of Coffee Production",
  "Access to Credit")

colnames <- c("Variable","Type", "SMD", "Balance Threshold", "Variance Ratio")

rownames(coffee_rep_btab) <- rowlabels

coffee_rep_btab[,3] <- ifelse(
          coffee_rep_btab[,3] >= "Not Balanced, >0.1", "No", "Yes")

kbl(coffee_rep_btab, digits=3, booktabs=TRUE, align="c",
```

Table 3.4: PALCHOLDER

Variable	Type	SMD	Balance Threshold	Variance Ratio
Household Age	Contin.	-0.272	No	1.073
Squared Household Age	Contin.	-0.255	No	1.143
Non-farm Income Access	Binary	0.301	No	NA
Log Total Land	Contin.	0.260	No	1.297
Dependency Ratio	Contin.	-0.400	No	0.979
Bad Weather	Binary	0.202	No	NA
Education Level	Contin.	0.244	No	1.034
Gender	Binary	-0.132	No	NA
Years of Coffee Production	Contin.	-0.340	No	0.911
Access to Credit	Binary	0.195	No	NA

```
    font_size=10, col.names=colnames) %>%
  kable_styling(full_width=TRUE)
```

```
# add render info for showtext to yaml. also chang legend to be more informative.
library(ggplot2)
love.plot(coffee_formula,
          data = coffee_data,
          weights = list(Replication = coffee_rep_pmodel),
          var.order = "unadjusted", binary = "std",
          abs = TRUE, colors = c("#333333", "#2780e3"),
          shapes = c("circle", "square"),
          line = TRUE, thresholds=0.1, s.d.denom="treated") +
  labs(title = "Variable Balance",
       x = "Absolute Standardised Mean Differences",
      fill="Method") +
  custom_ggplot_theme +
  scale_x_continuous(breaks = seq(0,0.6,length.out=7),expand = expansion(c(0, 0.05)))
```

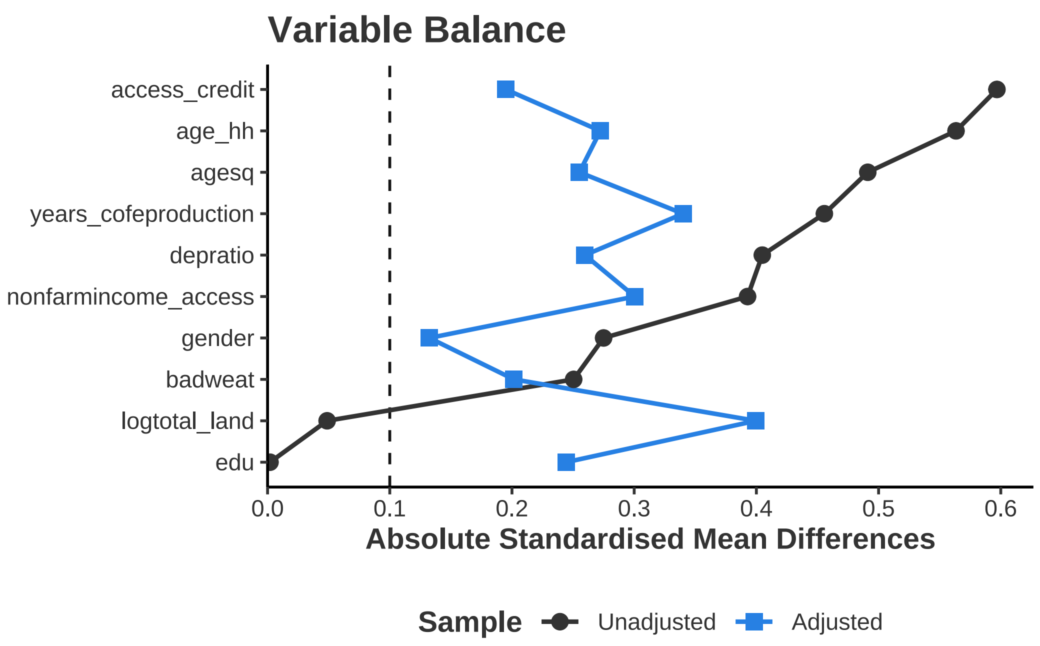


Figure 3.4: PLACEHOLDER

Table 3.4 and Figure 3.4 show that the propensity score matching process has obtained very poor balance. Based on the 10% rule, not a single variable is balanced and so the estimate of the treatment effect is likely to be biased by the structural differences in the groups.

For key variables such as Age, Gender, or Education, balance is especially important. On a theoretical level, we expect that people who are more educated are more likely to become certified as they are better able to engage with the application process and also are expected to earn more as increased education should lead to greater productivity. There likely exists gender discrimination given the time period and geographic area which suggests woman are less likely to be certified than men while also earning less due to a wide gender pay gap. These variables are strong confounders in theory and so emphasising balance in these variables is critical to making a robust causal inference.

Figure 3.5 shows trimmed regions that mostly impact the control group. Table 3.5 shows 34 observations are dropped of which 33 are treated and 1 are control being dropped. This increases balance as the trimmed observations are extreme data points. When observations are discarded, the ATT, ATC, or ATE cannot be claimed. Instead, this is refereed to as the average treatment effect on the matched or ATM. There is a significant reduction in the effective sample size as due to dropped obervations in the control group which has an effective sample size of 21 obervations down from 82.

```r
discarded_scores <- coffee_rep_pmodel$distance[coffee_rep_pmodel$discarded]


discard_min <- min(discarded_scores, na.rm = TRUE)
discard_max <- max(discarded_scores, na.rm = TRUE)


ggplot(coffee_data, aes(x = coffee_rep_pmodel$distance, fill = factor(certified))) +
  geom_density(alpha = 0.6, size = 0.6) +
  scale_fill_manual(values = c("#e5e5e5", "#2780e3"),
                    labels = c("Control", "Certified")) +
  labs(title = "Distribution of Propensity Scores in @coffeecite",
       x = "Propensity Scores", y = "Density", fill = "Group:") +
  scale_x_continuous(expand = expansion(0), limits = c(0,1)) +
  scale_y_continuous(expand = expansion(0), limits = c(0,5)) +
  geom_vline(xintercept = discard_min, color = "#333333", size = 0.8) +
  geom_vline(xintercept = discard_max, color = "#333333", size = 0.8) +
  annotate("rect", xmin = 0, xmax = discard_min, ymin = -Inf, ymax = Inf, fill = "#333333", a
  annotate("rect", xmin = discard_max, xmax = 1, ymin = -Inf, ymax = Inf, fill = "#333333", a
  annotate("text", x = 0.02, y = 2.5,
           label = "Discarded Range", angle = 90, vjust = 1.5, size=4,fontface="bold", color
  custom_ggplot_theme
```

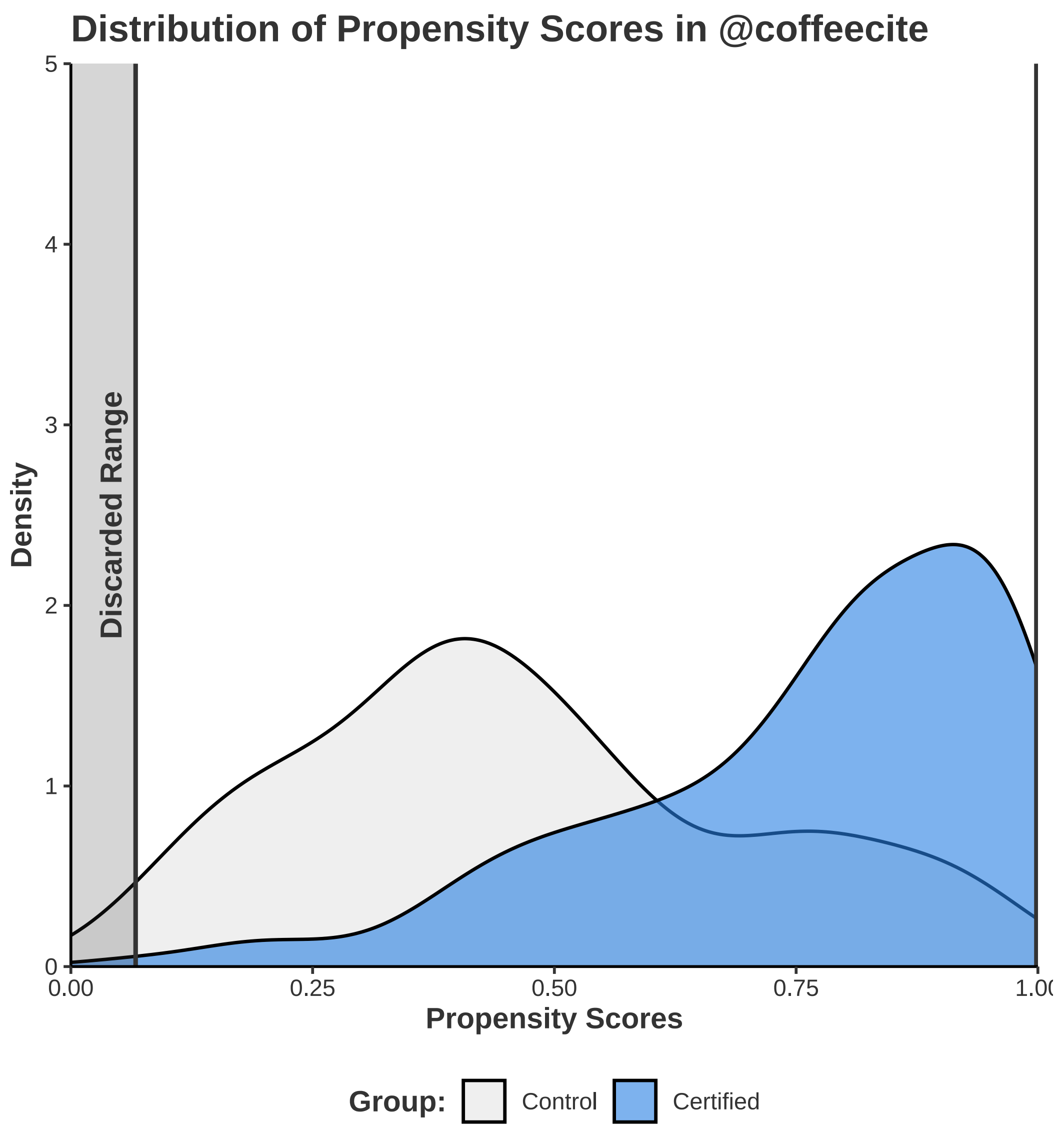


Figure 3.5: PLACEHOLDER

Table 3.5: Placeholder

	Control	Treated
All (ESS)	82	164
All (Unweighted)	82	164
Matched (ESS)	21	131
Matched (Unweighted)	42	131
Unmatched	39	0
Discarded	1	33

```
kbl(coffee_rep_btab_ss, digits=0, booktabs=TRUE, align="c",
    font_size=10) %>%
  kable_styling(full_width=F)
```

Overall, this model fit using logistic regression and propensity score matching has resulted in a poor model due to covariate imbalance and unidentifiable estimands. It is likely that improvement can be made using

3.5.2 Further Modelling

In the following model fitting process, I aim to obtain better results while preserving the estimand. To improve the poor balance achieved by the paper there are two strategies to obtain better balance. First, the propensity scores can be re-estimated using machine learning leading to better calibrated propensity scores. Second, inverse propensity weighting (IPW) can be used instead of propensity score matching (PSM). IPW should ensure that that the sample size remains the same as no observations are lost in a matching process. Additionally, IPW is generally more efficient as the pseudo-population is based on prescise weights compared to matching that is based on approximate similarity between treatment and control.

The machine learning propensity scores will be estimated using `WeightIt` in the same process as **?@sec-demo**. To select the criteria that defines the best model, consider Figure 3.4 that shows there is a significant range of balance levels in the raw dataset. Knowing this, the model will be tuned using `criterion = "smd.max"` as reducing the extremely unbalanced variables is important to achieving balance even if this leads to a higher average standardised mean difference (SMD). Additionally, the model fitting process was completed using `criterion = "smd.max"` and is shown in **?@sec-appendix**.

```r
library(WeightIt)
library(cobalt)

set.seed(88)
coffee_boosted_weight <- weightit(coffee_formula, data=coffee_data,
                                  method="gbm", distribution="bernoulli",
                                  use.offset=c(T),
                                  shrinkage=seq(0.15, 0.4,length.out=5),
                                  bag.fraction=0.67,
                                  interaction.depth=3:6,
                                  n.trees=500,
                                  criterion="smd.mean",
                                  estimand="ATT")



coffee_boosted_btab <- bal.tab(coffee_boosted_weight,
                         data = coffee_data,
                         stats = c("mean.diffs","variance.ratios"),
                         binary = "std", continuous = "std",
                         thresholds = c(mean.diffs = 0.1),
                         s.d.denom = "treated")

# Extracts the balance tabltune# Extracts the balance table and removes unwanted columns.
coffee_boosted_btab <- coffee_boosted_btab$Balance[-1,-c(2,3)]
```

Discussion of Tuning

Initially, a tuning grid considering shrinkage values of $0.001, 0.005, .01, 0.05, 0.1,$ and 0.2 were considered using 10000 trees with a depth between 1 and 5. The best tuning performance was found with shrinkage of 0.2 and 9 trees which were three splits 3 deep. As such, the tuning grid was redefined in a second iteration to use $0.1, 0.15, 0.2, 0.25, 0.3, 0.35,$ and 0.4 with only 1000 trees with between 2 and 5 depth. The second fit, suggested a learning rate of 0.35 so the local area of $0.3, 0.325, 0.350, 0.375,$ and 0.4 is searched in the final fit.

Of course there is no guarantee that the GBM model will perform the best and so a logisic model is also fitted. An interesting comparison is between the balance visible in the matched sample like in (**coffeecite?**) and in the weighted sample. Any difference between these two samples relates to the difference between matching and weighting on the propensity score as the score is the same in both methods (generated by logistic regression).

```
coffee_logit_weight <- weightit(coffee_formula, data=coffee_data, method="glm",
                                        estimand="ATT")

coffee_logit_btab <- bal.tab(coffee_logit_weight,
                                     formula = coffee_formula,
                                     data = coffee_data,
                                     stats = c("mean.diffs","variance.ratios"),
                                     binary = "std", continuous = "std",
                                     thresholds = c(mean.diffs = 0.1),
                                     s.d.denom = "treated")

coffee_logit_btab <- coffee_logit_btab$Balance[-1,-c(2,3)]
```

Additionally, the balance present in the raw data is calculated.

```
coffee_raw_btab <- bal.tab(coffee_formula,
                         data = coffee_data,
                         stats = c("mean.diffs","variance.ratios"),
                         binary = "std", continuous = "std",
                         thresholds = c(mean.diffs = 0.1),
                         s.d.denom = "treated")

coffee_raw_btab <- coffee_raw_btab$Balance[,-c(5,6)]
```

3.5.3 Comparison of Methods

In each of the above code chunks, the `cobalt` table has computed balance tables which are combined together in Table 3.6 and Figure 3.6.

```
library("data.table")

coffee_combined_btab <- rbindlist(list(coffee_raw_btab,
                                       coffee_logit_btab,
                                       coffee_boosted_btab), use.names=FALSE)

coffee_combined_btab$Variable <- rep(rowlabels,3)

coffee_combined_btab <- coffee_combined_btab[,c(5,1,2,3,4)]

coffee_combined_btab[,4] <- ifelse(
          coffee_combined_btab[,4] >= "Not Balanced, >0.1", "No", "Yes")
```

```
library(kableExtra)
kbl(coffee_combined_btab, digits=3, booktabs=TRUE, align="c",
    font_size=10, col.names=colnames) %>%
  kable_styling(full_width=TRUE) %>%
  column_spec(1, bold=TRUE, width="5cm") %>%
  column_spec(2:5, bold=FALSE, width="1cm") %>%
  pack_rows("Raw Data", 1, 10, label_row_css = "text-align: center;") %>%
  pack_rows("Logistic Regression and IPTW", 11, 20, label_row_css = "text-align: center;") %
  pack_rows("Boosted Machine with IPTW", 21, 30, label_row_css = "text-align: center;")
```

```
love.plot(coffee_formula,
          data = coffee_data,
          weights = list(Replication = coffee_rep_pmodel,
                         Logit = coffee_logit_weight,
                         Boosting= coffee_boosted_weight),
          var.order = "unadjusted", binary = "std",continuous = "std",
          abs = TRUE, colors = c("#333333", "#2780e3", "darkblue","darkred"),
```

Table 3.6: Comparison of Balance for Coffee Data Using Different Propensity Models

Variable	Type	SMD	Balance Threshold	Variance Ratio
Raw Data				
Household Age	Contin.	0.563	No	0.865
Squared Household Age	Contin.	0.491	No	1.007
Non-farm Income Access	Binary	-0.393	No	NA
Log Total Land	Contin.	-0.405	No	0.551
Dependency Ratio	Contin.	0.049	Yes	1.237
Bad Weather	Binary	-0.250	No	NA
Education Level	Contin.	-0.002	Yes	0.727
Gender	Binary	-0.275	No	NA
Years of Coffee Production	Contin.	0.456	No	1.362
Access to Credit	Binary	0.597	No	NA
Logistic Regression and IPTW				
Household Age	Contin.	0.245	No	0.927
Squared Household Age	Contin.	0.228	No	1.072
Non-farm Income Access	Binary	0.170	No	NA
Log Total Land	Contin.	-0.092	Yes	0.856
Dependency Ratio	Contin.	0.114	No	1.388
Bad Weather	Binary	0.194	No	NA
Education Level	Contin.	0.047	Yes	0.922
Gender	Binary	-0.046	Yes	NA
Years of Coffee Production	Contin.	-0.061	Yes	1.112
Access to Credit	Binary	-0.029	Yes	NA
Boosted Machine with IPTW				
Household Age	Contin.	0.067	Yes	1.269
Squared Household Age	Contin.	0.099	Yes	1.491
Non-farm Income Access	Binary	0.058	Yes	NA
Log Total Land	Contin.	-0.028	Yes	0.876
Dependency Ratio	Contin.	-0.062	Yes	0.766
Bad Weather	Binary	0.191	No	NA
Education Level	Contin.	0.138	No	1.073
Gender	Binary	-0.082	Yes	NA
Years of Coffee Production	Contin.	-0.006	Yes	0.970
Access to Credit	Binary	0.123	No	NA

```
        shapes = c("circle", "square", "triangle", "diamond"),
        line = TRUE,thresholds=0.1,s.d.denom="treated",use.grid=F)+
  labs(title = "Variable Balance Using Different Balance Methods",
       x = "Absolute Standardised Mean Differences",
       fill="Method") +
  scale_x_continuous(breaks = seq(0,0.6,length.out=7),expand = expansion(c(0, 0.05))) + custo
```

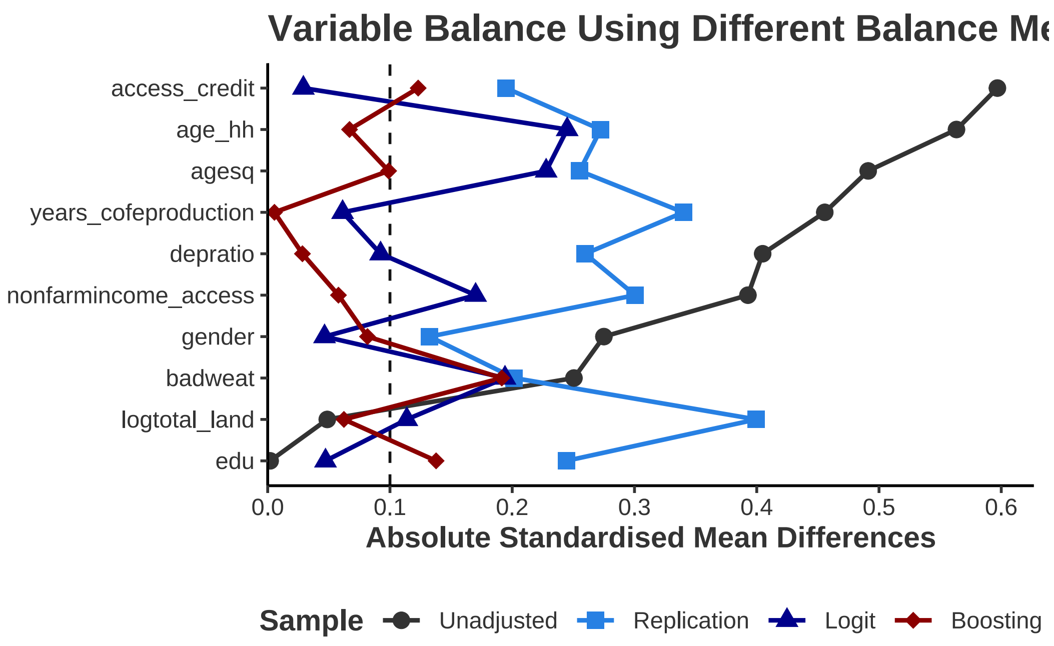


Figure 3.6: Comparison of Balance for Coffee Data Using Different Methods

Viewing results of our balance shows three notable findings:

1. PSM has performed very poorly relative to IPW even though matching dropped a significant number of observations.

2. A GBM model has resulted in better covariate balance than logistic regression for most variables. Using a 10% threshold for determining balance, logistic regression leaves 5 vari-

ables unbalanced and the GBM leaves 3 variables unbalanced. Additionally, the margin above being unbalanced is also larger for logistic regression. Although logistic regression results in better balance for “access to credit”, “gender”, and “education”. Logistic regress has the average balance across all variables is 0.0768603 which is satisfactory. For boosting, The average standardised mean is 0.0498114 which is quite impressive compared to the methodology used in the paper.

3. The variable with the worst balance for each model is Age with a SMD of 0.245 for logistic regression and “bad weather” with 0.191 for the GBM.

4. Using a looser balance threshold of 0.2, the GBM achieves balance across all covariates while the logistic regression does not balance either of the Age variables.

For the logistic regression model, the balance statistics are marginally balanced. Using a 10% threshold, half of the variables are balanced. Using a relaxed 20% threshold, only Age and Age Squared are unbalanced but balance with threshold should be interpreted with caution.

Now that satisfactory covariate balance is achieved, the treatment effect can be estimated under logistic regression, the GBM, and then compared to the result in the paper. Note that the estimand in the paper is intended to be the average treatment effect (ATT) but dropped observations mean the actual treatment effect is the average treatment effect on matched (ATM) individuals.

```
coffee_att_formula <- update.formula(as.formula(paste("~", paste(attr(terms(coffee_formula),



coffee_logit_fit <- lm_weightit(coffee_att_formula,
                       data = coffee_data, weightit = coffee_logit_weight)
```

Table 3.7: PLACEHOLDER

	Estimate	SE	P.Value	Lower.CI	Upper.CI
Rep. Result (Logistic with PSM)	-0.1538	0.7384	0.8350	-1.6009	1.2934
Logistic Regression and IPW	-1.5824	0.6072	0.0092	-2.7724	-0.3924
Generalized Boosting Machine and IPW	-1.0187	0.5196	0.0499	-2.0372	-0.0003

```r
coffee_boosted_fit <- lm_weightit(coffee_att_formula,
                                 data = coffee_data, weightit = coffee_boosted_weight)



coffee_logit_att <- avg_comparisons(coffee_logit_fit, variables = "certified")


coffee_boosted_att <- avg_comparisons(coffee_boosted_fit, variables = "certified")


coffee_comparisons_tab <- rbind(replicated_result_tbl, extract_comparison_results(coffee_log
                             extract_comparison_results(coffee_boosted_att))
rownames(coffee_comparisons_tab) <- c("Rep. Result (Logistic with PSM)","Logistic Regression


kbl(coffee_comparisons_tab, digits=4,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
```

3.6 Code Provided for PDF Output

```r
load(file = "my_environment.RData")
library(randomForest)
set.seed(88)
nsw_formula <- as.formula(as.factor(treat) ~ age + educ + re75 +
                          black + hisp + degree + marr)

logit_preds <- glm(nsw_formula, data = nsw_data,
                   family = binomial())$fitted.values

rf_mtry1_preds <- predict(randomForest(nsw_formula,
                          mtry = 1, data = nsw_data),
                          newdata = nsw_data, type = "prob")[, 2]

bagging_model <- randomForest(nsw_formula, mtry = 7, importance = TRUE,
                              data = nsw_data)

bagged_preds <- predict(bagging_model, newdata = nsw_data, type = "prob")[, 2]

library(ggplot2)
plot_pmachines <- function(pscores, plot_subtitle) {
  ggplot(nsw_data, aes(x = pscores, fill = factor(treat))) +
    geom_density(alpha = 0.6, size = 0.6) +
    scale_fill_manual(values = c("#e5e5e5", "#2780e3"),
                      labels = c("Control", "Participant")) +
    labs(subtitle = plot_subtitle, x = "Propensity Scores", y = "Density",
         fill = "Group:") +
    scale_x_continuous(expand = expansion(0), limits = c(0,1)) +
    scale_y_continuous(expand = expansion(0), limits = c(0,10)) +
    custom_ggplot_theme
}

p1 <- plot_pmachines(logit_preds, "Logistic Regression") + xlab(NULL) +
  theme(legend.position = "none") +
  annotate(geom = "curve", x = 0.6, y = 5, xend = 0.42, yend = 0,
           curvature = .3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 0.6, y = 5, label = "True Probability",
           hjust = "left", color = "#333333", size = 3,
           family = "Source Sans Pro")

p2 <- plot_pmachines(rf_mtry1_preds, "Random Forest (mtry = 1)") + xlab(NULL) +
  theme(legend.position = "none")
p3 <- plot_pmachines(bagged_preds, "Bagging (Bootstrap Aggregation)")

library(patchwork)
p1 / p2 / p3 + plot_annotation(
  title = "Density Plots of Propensity Scores for NSW Data")
library(ggplot2)
library(tidyverse)
imp <- as.data.frame(importance(bagging_model))
imp <- cbind(vars = rownames(imp), imp)
imp <- imp[order(imp$MeanDecreaseGini),]
imp$vars <- factor(imp$vars, levels = unique(imp$vars))

imp %>%
  pivot_longer(cols = matches("Mean")) %>%
  ggplot(aes(y = vars, x = value, fill = name)) +
  geom_bar(stat = "identity", width = 0.8, show.legend = TRUE,
           position = position_dodge(width = 0.8), color = "black", size = 0.6) +
  facet_grid(~ factor(name, levels = c("MeanDecreaseGini", "MeanDecreaseAccuracy")), scales =
  scale_fill_manual(values = c("#e5e5e5", "#2780e3")) +
  scale_x_continuous(expand = expansion(c(0, 0.04))) +
  labs(
    title = "Variable Importance",
    x = "% Decrease if Variable is Omitted from Model",
    y = "Variable Name"
  ) + custom_ggplot_theme +
  theme(
    legend.position = "none"
  )

library(WeightIt)
nsw_logit_weight <- weightit(nsw_formula, data = nsw_data,                   ①
                             ps = nsw_logit_pscores,                         ②
                             estimand = "ATE")                               ③
set.seed(88)
nsw_boosted_weight2 <- weightit(nsw_formula, data = nsw_data,
                               method="gbm",
                               estimand = "ATE",
                               shrinkage= c(0.25, 0.3, 0.35, 0.4, 0.45, 0.5),
                               interaction.depth = 1:3,
                               bag.fraction = 1,
                               offset = c(TRUE, FALSE),
                               criterion = "smd.mean",
                               n.trees = 5000)

print(nsw_boosted_weight2$info$best.tune)
low_shrinkage <- weightit(nsw_formula, data = nsw_data,
                               method = "gbm",
                               estimand = "ATE",
                               shrinkage = 0.05,
                               interaction.depth = 1,
                               offset = c(TRUE, FALSE),
                               criterion = "smd.mean",
                               n.trees = 40000)

library(ggplot2)
optimal_boost_plot <- ggplot(nsw_boosted_weight2$info$tree.val, aes(x = tree, y = smd.mean))
  geom_line(size = 1, color = "#2780e3") +
  labs(subtitle = "Optimal Tune",
       x = "Number of Iterations",
       y = "Average Standardised Mean Difference") +
  custom_ggplot_theme +
  xlim(0,500)


lowshrinkage_boost_plot <- ggplot(low_shrinkage$info$tree.val, aes(x = tree, y = smd.mean)) 
  geom_line(size = 1, color = "#2780e3") +
  labs(subtitle = "Low Learning Rate (shrinkage = 0.05)",
       x = "Number of Iterations",
       y = NULL) +
  custom_ggplot_theme +
  annotate(geom = "curve", x = 30000, y = 0.05,
           xend = low_shrinkage$info$best.tree, yend = 0.0231,
           curvature = 0.3, arrow = arrow(length = unit(2, "mm"))) +
  annotate(geom = "text", x = 31000, y = 0.05, label = "Minimum",
           hjust = "left", color = "#333333", size = 3, family = "Source Sans Pro")


optimal_boost_plot + lowshrinkage_boost_plot + plot_annotation(
  title = 'Number of Tree Iterations and Balance')
nsw_boosted_btab <- bal.tab(nsw_boosted_weight,
                            data = nsw_data,
                            stats = c("mean.diffs","variance.ratios"),
                            binary = "std", continuous = "std",
                            thresholds = c(mean.diffs = 0.1))


nsw_raw_btab <- bal.tab(nsw_formula,
                        data = nsw_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
```

```r
                         thresholds = c(mean.diffs = 0.1),
                         s.d.denom = "treated")

# Extracts the balance table and removes unwanted columns.
nsw_boosted_btab <- nsw_boosted_btab$Balance[-1,-c(2,3)]
nsw_raw_btab <- nsw_raw_btab$Balance[-c(5,6)]
library(dplyr)
library(kableExtra)

collabels <- c("Type", "SMD", "Balanced", "Variance Ratio","Method")

rowlabels <- c("Age", "Education", "Income 1975","Black",
               "Hispanic", "Degree", "Married")

nsw_raw_btab$method <- "Raw Data"
nsw_logit_btab$method <- "IPTW: Logistic Regression"
nsw_boosted_btab$method <- "IPTW: Boosting"

combined_btab <- bind_rows(setNames(nsw_raw_btab,collabels),
                           setNames(nsw_logit_btab,collabels),
                           setNames(nsw_boosted_btab,collabels))

combined_btab$Variable <- rep(rowlabels,3)

combined_btab <- combined_btab[c(6,1,2,3,4,5)]
```

```r
rownames(combined_btab) <- NULL

combined_btab$Balanced <- ifelse(
          combined_btab$Balanced == "Not Balanced, >0.1", "No", "Yes")

kbl(combined_btab[-6], digits=2,booktabs= T,align = "c",
      font_size=10) %>%
  kable_styling(full_width = T) %>%
  row_spec(0, bold = TRUE) %>%
  column_spec(1, bold = TRUE) %>%
  column_spec(2:5, bold = F, width="3cm") %>%
  pack_rows(index = rev(table(combined_btab$Method)))
nsw_logit_lm <- lm_weightit(re78~treat*(age + educ +
                             re75 + black + hisp +
                             degree + marr), data = nsw_data,
                             weights = nsw_logit_weight$weights)

nsw_logit_result <- avg_comparisons(nsw_logit_lm, variables = "treat")

nsw_comparisons_tab <- rbind(extract_comparison_results(nsw_logit_result),
                             extract_comparison_results(nsw_boosted_result))
rownames(nsw_comparisons_tab) <- c("Logistic Regression", "GBM")

kbl(nsw_comparisons_tab, digits=2,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
```

```r
nsw_logit_lm <- lm_weightit(re78~treat*(age + educ +
                             re75 + black + hisp +
                             degree + marr), data = nsw_data,
                             weights = nsw_logit_weight$weights)


nsw_logit_result <- avg_comparisons(nsw_logit_lm, variables = "treat")


nsw_comparisons_tab <- rbind(extract_comparison_results(nsw_logit_result),
                             extract_comparison_results(nsw_boosted_result))
rownames(nsw_comparisons_tab) <- c("Logistic Regression", "Generalized Boosting Machine ")


kbl(nsw_comparisons_tab, digits=2,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
coffee_formula <- as.formula(certified ~ age_hh +
                  agesq + nonfarmincome_access + depratio +
                  logtotal_land + badweat + edu + gender +
                  years_cofeproduction + access_credit)


library(MatchIt)
library(marginaleffects)
coffee_rep_pmodel <- matchit(coffee_formula, data=coffee_data, distance="glm",
                              method="nearest", replace = T, estimand="ATT",
                              discard="both")


coffee_logit_md <- match.data(coffee_rep_pmodel)
```

```r
coffee_rep_fit<- lm(percapitaincome_day_maleeq ~ certified, data = coffee_logit_md, weights=

replicated_result <- avg_comparisons(coffee_rep_fit, variables = "certified",
                vcov = TRUE,
                newdata = subset(coffee_logit_md, certified == 1),
                wts = "weights")
replicated_result_tbl <- extract_comparison_results(replicated_result)
rownames(replicated_result_tbl) <- "Replicated Result"
kbl(replicated_result_tbl, digits=2,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
library(cobalt)
coffee_rep_btab <- bal.tab(coffee_rep_pmodel,
                        data = coffee_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
                        thresholds = c(mean.diffs = 0.1),
                        s.d.denom = "treated")

coffee_rep_btab_ss <- coffee_rep_btab$Observations

coffee_rep_btab <- coffee_rep_btab$Balance[-1,-c(2,3)]

rowlabels <- c(
  "Household Age", "Squared Household Age", "Non-farm Income Access",
  "Log Total Land", "Dependency Ratio", "Bad Weather",
  "Education Level", "Gender", "Years of Coffee Production",
  "Access to Credit")

colnames <- c("Variable","Type", "SMD", "Balance Threshold", "Variance Ratio")

rownames(coffee_rep_btab) <- rowlabels

coffee_rep_btab[,3] <- ifelse(
          coffee_rep_btab[,3] >= "Not Balanced, >0.1", "No", "Yes")

kbl(coffee_rep_btab, digits=3, booktabs=TRUE, align="c",
    font_size=10, col.names=colnames) %>%
  kable_styling(full_width=TRUE)
nobs_coffee_dropped <- sum(coffee_rep_pmodel$discarded, na.rm=TRUE)
coffee_data$discarded <- coffee_rep_pmodel$discarded
nobs_coffee_Tdropped <- nrow(subset(coffee_data,discarded==TRUE&certified==1))
nobs_coffee_Cdropped <- nrow(subset(coffee_data,discarded==TRUE&certified==0))
# add render info for showtext to yaml. also chang legend to be more informative.
library(ggplot2)
love.plot(coffee_formula,
          data = coffee_data,
          weights = list(Replication = coffee_rep_pmodel),
          var.order = "unadjusted", binary = "std",
          abs = TRUE, colors = c("#333333", "#2780e3"),
          shapes = c("circle", "square"),
          line = TRUE, thresholds=0.1, s.d.denom="treated") +
  labs(title = "Variable Balance",
       x = "Absolute Standardised Mean Differences",
       fill="Method") +
  custom_ggplot_theme +
  scale_x_continuous(breaks = seq(0,0.6,length.out=7),expand = expansion(c(0, 0.05)))

discarded_scores <- coffee_rep_pmodel$distance[coffee_rep_pmodel$discarded]

discard_min <- min(discarded_scores, na.rm = TRUE)
discard_max <- max(discarded_scores, na.rm = TRUE)

ggplot(coffee_data, aes(x = coffee_rep_pmodel$distance, fill = factor(certified))) +
  geom_density(alpha = 0.6, size = 0.6) +
  scale_fill_manual(values = c("#e5e5e5", "#2780e3"),
                    labels = c("Control", "Certified")) +
  labs(title = "Distribution of Propensity Scores in @coffeecite",
       x = "Propensity Scores", y = "Density", fill = "Group:") +
  scale_x_continuous(expand = expansion(0), limits = c(0,1)) +
  scale_y_continuous(expand = expansion(0), limits = c(0,5)) +
  geom_vline(xintercept = discard_min, color = "#333333", size = 0.8) +
  geom_vline(xintercept = discard_max, color = "#333333", size = 0.8) +
  annotate("rect", xmin = 0, xmax = discard_min, ymin = -Inf, ymax = Inf, fill = "#333333", a
  annotate("rect", xmin = discard_max, xmax = 1, ymin = -Inf, ymax = Inf, fill = "#333333", a
  annotate("text", x = 0.02, y = 2.5,
           label = "Discarded Range", angle = 90, vjust = 1.5, size=4,fontface="bold", color
  custom_ggplot_theme
kbl(coffee_rep_btab_ss, digits=0, booktabs=TRUE, align="c",
    font_size=10) %>%
  kable_styling(full_width=F)
library(WeightIt)
library(cobalt)

set.seed(88)
coffee_boosted_weight <- weightit(coffee_formula, data=coffee_data,
                                method="gbm", distribution="bernoulli",
                                use.offset=c(T),
                                shrinkage=seq(0.15, 0.4,length.out=5),
                                bag.fraction=0.67,
                                interaction.depth=3:6,
                                n.trees=500,
                                criterion="smd.mean",
                                estimand="ATT")




coffee_boosted_btab <- bal.tab(coffee_boosted_weight,
                        data = coffee_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
                        thresholds = c(mean.diffs = 0.1),
                        s.d.denom = "treated")
```

```r
# Extracts the balance tabltune# Extracts the balance table and removes unwanted columns.
coffee_boosted_btab <- coffee_boosted_btab$Balance[-1,-c(2,3)]
coffee_logit_weight <- weightit(coffee_formula, data=coffee_data, method="glm",
                                estimand="ATT")

coffee_logit_btab <- bal.tab(coffee_logit_weight,
                             formula = coffee_formula,
                             data = coffee_data,
                             stats = c("mean.diffs","variance.ratios"),
                             binary = "std", continuous = "std",
                             thresholds = c(mean.diffs = 0.1),
                             s.d.denom = "treated")

coffee_logit_btab <- coffee_logit_btab$Balance[-1,-c(2,3)]
coffee_raw_btab <- bal.tab(coffee_formula,
                        data = coffee_data,
                        stats = c("mean.diffs","variance.ratios"),
                        binary = "std", continuous = "std",
                        thresholds = c(mean.diffs = 0.1),
                        s.d.denom = "treated")

coffee_raw_btab <- coffee_raw_btab$Balance[,-c(5,6)]
library("data.table")

coffee_combined_btab <- rbindlist(list(coffee_raw_btab,
                                        coffee_logit_btab,
                                        coffee_boosted_btab), use.names=FALSE)


coffee_combined_btab$Variable <- rep(rowlabels,3)


coffee_combined_btab <- coffee_combined_btab[,c(5,1,2,3,4)]


coffee_combined_btab[,4] <- ifelse(
          coffee_combined_btab[,4] >= "Not Balanced, >0.1", "No", "Yes")
library(kableExtra)
kbl(coffee_combined_btab, digits=3, booktabs=TRUE, align="c",
    font_size=10, col.names=colnames) %>%
  kable_styling(full_width=TRUE) %>%
  column_spec(1, bold=TRUE, width="5cm") %>%
  column_spec(2:5, bold=FALSE, width="1cm") %>%
  pack_rows("Raw Data", 1, 10, label_row_css = "text-align: center;") %>%
  pack_rows("Logistic Regression and IPTW", 11, 20, label_row_css = "text-align: center;") %
  pack_rows("Boosted Machine with IPTW", 21, 30, label_row_css = "text-align: center;")
love.plot(coffee_formula,
          data = coffee_data,
          weights = list(Replication = coffee_rep_pmodel,
                         Logit = coffee_logit_weight,
                         Boosting= coffee_boosted_weight),
          var.order = "unadjusted", binary = "std",continuous = "std",
          abs = TRUE, colors = c("#333333", "#2780e3", "darkblue","darkred"),
          shapes = c("circle", "square", "triangle", "diamond"),
          line = TRUE,thresholds=0.1,s.d.denom="treated",use.grid=F)+
  labs(title = "Variable Balance Using Different Balance Methods",
       x = "Absolute Standardised Mean Differences",
       fill="Method") +
  scale_x_continuous(breaks = seq(0,0.6,length.out=7),expand = expansion(c(0, 0.05))) + custo
coffee_att_formula <- update.formula(as.formula(paste("~", paste(attr(terms(coffee_formula),



coffee_logit_fit <- lm_weightit(coffee_att_formula,
                    data = coffee_data, weightit = coffee_logit_weight)


coffee_boosted_fit <- lm_weightit(coffee_att_formula,
                                 data = coffee_data, weightit = coffee_boosted_weight)



coffee_logit_att <- avg_comparisons(coffee_logit_fit, variables = "certified")


coffee_boosted_att <- avg_comparisons(coffee_boosted_fit, variables = "certified")


coffee_comparisons_tab <- rbind(replicated_result_tbl, extract_comparison_results(coffee_logi
                             extract_comparison_results(coffee_boosted_att))
rownames(coffee_comparisons_tab) <- c("Rep. Result (Logistic with PSM)","Logistic Regression


kbl(coffee_comparisons_tab, digits=4,booktabs= T, align = "c",
      font_size=10) %>%
  kable_styling(full_width = T)
```

```r
save.image(file = "my_environment.RData")
```

4 Meta-Learners

Künzel et al. (2019) offers a summary of S-Learners, T-Learners and introduces X-Learners. Each of these methods offers a methodology of increasing complexity to estimate heterogeneous treatment effects using any type of base learner.

In this case a base learner is the machine learning used to in the metalearning framework. This is commonly a tree-based method such as Random Forest. As discussed in figure, the base-learner itself is not adapted to a causal context. Instead, the way the models are implemented together will explore causality.

Note than in each of these metalearners the outcome variable can be binary or continuous. In a binary case, the base-learner will be any learner cabable of classification. In a contineous case, the base-learner will be any learner capable of regression.

4.1 What is a Base-learner?

4.2 S-Learners

Single learners or S-Learner uses a single base learner, M_s, where the treatment indicator is given no special status in the model. Simply, the model is a base-learner with a dependant variable where the treatment indicator is incorporated alongside other covariates without being given any unique status or special treatment within the model structure.

The single base-learner will model the feature space and how the covariates (including the treatment) predict the outcome. Recall from Equation 2.1 that the ITE is the difference between the two potential outcomes. The inference of causality in the S-learner is that we use the predictive power of machine learning to predict each of the potential outcomes. The predicted CATE for an individual unit is then the difference between the predicted outcomes when the treatment assignment indicator is changed from control to treatment.

Intuition of S-Learners

The intuition of how meta-learners work is best expressed visually using a simple CART model as in **?@fig-s__learner__process** because we can interpret the splits. When predicting outcomes from covariates and treatment, the tree algorithm splits the data to form more homogeneous nodes, making the nodes purer than before the split. In doing so, CART may split on the treatment indicator at various points within the tree. Treatment effect heterogeneity is captured by splitting the homogeneous nodes on the treatment indicator, resulting in different treatment effects across the resulting nodes. Essentially, the tree structure illustrates how the treatment effect varies by considering the treatment variable across different homogeneous groups, thereby highlighting the heterogeneity in the treatment effect across the population. When making predictions from the tree for each potential outcome, the location of the splits explains the heterogeneity of the effect.

Assuming a binary treatment, the individual treatment effect can be predicted as:

$$\hat{\delta}_i = y_i^{pred}(1) - y_i^{pred}(0) \tag{4.1}$$

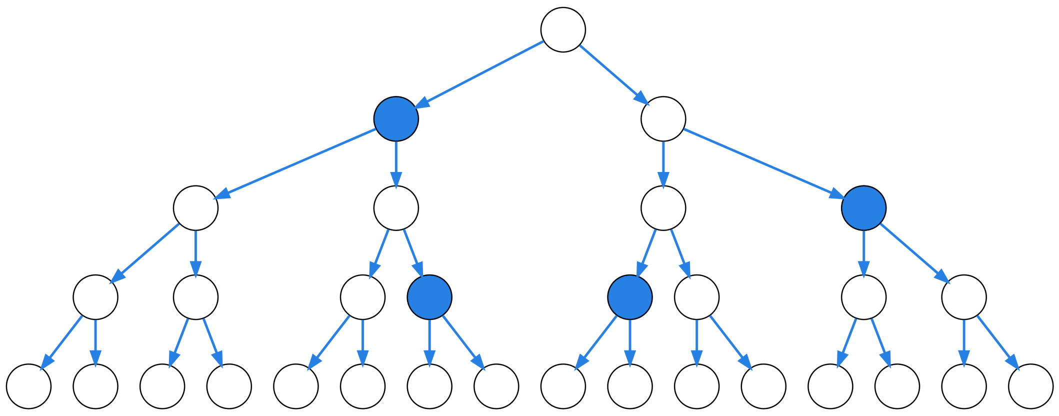


Figure 4.1: S-learner Visualisation. Blue shaded nodes represent splits on the treatment indicator.

A notable property of the S-Learner is that the base-learner may not consider the treatment effect for each individual. This can occur becuase the CATE may be zero for some individuals or because the treatment indicator may be quite *weak*. An indicator may be weak because the effect is small or noisy. In **?@fig-s_learner_process**, then means that the tree does not split on the treatment indicator. When assessing each split, at no point does splitting on the treatment result in the most purity in a node.

Given that there indicator may not be considered, it makes sense that the estimate of the treatment effect would be biased twoards zero. Künzel et al. (2019) perform simulations and show that the S-Learner tends to bias the treatment towards zero as expected.

Another notable property is that S-Learners will model the response surface of the treatment and control groups together as we only have a single base-learner. This means that the base-learner models the data as if the treatment and control groups share a similar response surface. This means the functional form that maps features to outcomes is similar across both groups. Conceptually, the base-learner leverages information from both groups and develops a composite response surface. In other words, *the structure of the treatment group tells us about the structure of the control group, vise-versa.* Of course, this will not always be the case

- perhaps the relationship between variables in the control group are additive and suggesting a linear model is appropriate but the treatment group has multiplicative relationships and complex functional forms.

The above property can be beneficial or harmful depending on context. If prior domain knowledge suggests that the CATE may be zero in many observations, then the S-Learner may be a good candidate during model selection. Also, when we know that there is a similar response surface between groups then we can take advantage of this with a single base-learner using an S-Learner. Alternatively, there may prior knowledge suggests that there is a widely experienced treatment effect or we know that there are structural differences between each group, then the S-Learner is counter-intuitive and other models will likely perform better.

For illustrative purposes, consider the NSW jobs programme. Further information about this dataset can be found in **?@sec-data__nsw__jobs**. For conceptual understanding, a simple CART model is excellent however more complex models provide superior performance. One such model is a Random Forest which is implemented below using K-fold cross validation and a grid search procedure for `mtry`. I do not discuss these methods in depth beyond their implementation but there are good sources to learn about them for unfamiliar readers.

I use a test-train split as it is of interest to see how well the model predicts the change in income. Note that we cannot see how well the model predicts the resulting ITE's because there is no observable ITE. A purely theoretical comfort is that if a model predicts the known outcome well and the treatment and control group have similar response surfaces, then the ITE's should be robust.

```
library(caret)                                                    ①
library(randomForest)
```

```r
slearner_model_control <- trainControl(method = "cv", number=10)                  ②


Grid_rf <-  expand.grid(mtry = c(3,4,5,6))


slearner_rf <- train(re78~
                        treat + age + educ +
                        black + hisp + marr + nodegree,
                        data = nsw_data,
                        method = "rf",
                        trControl = slearner_model_control,
                        tuneGrid = rfGrid,
                        verbose=FALSE)                                            ③


y_hat1 <- predict(slearner_rf,
                  newdata=within(nsw_data,treat <- 1))                            ④


y_hat0 <- predict(slearner_rf,
                  newdata=within(nsw_data,treat <- 0))


slearner_ite_nsw <- y_hat1 - y_hat0                                               ⑤
```

① Install the `caret` and `randomForest` packages in `R`.

② Define the model control and hyperparameter tuning grid. Here the model control is K-fold cross validation with 10 folds and values 4 different values of `mtry` are considered.

③ The Random Forest model is *trained* and a grid search is performed for values of `mtry`. The `train` function is from the `caret` package and implements a grid search procedure and cross-validation.

④ Predictions of potential outcomes are made holding the treatment equal to 1 and 0. The `within()` function allows the specification of values for the `newdata`.

⑤ The ITE is the contrast of the predicted potential outcomes as in Equation 4.1.

Discuss further: - what to do with ITE, follow ups - what base to use - an example of s learner implementation in published research - confounding effects.

4.3 T-Learners

It would seem that the S-Learner is inadequate in many applications which the T-Learner aims to resolve. In the T-Learner there are two base learners - hence the T-Learner- each fit on treatment and control observations resulting in M_1 and M_0 respectively. The dependant variable remains the same in both. This results in two models that seperately models the outcome for each of the treatment and control cases separately. T-learners aim to answer the *fundamental problem* by using two separate base-learners associated with treatment and control to predict potential outcomes.

$$\begin{aligned} M_0 &= E[Y(0)|X = x, T = 0] \\ M_1 &= E[Y(1)|X = x, T = 1] \end{aligned} \tag{4.2}$$

Thus, predictions are then made from each base learner on the whole data resulting in counterfactual predictions. In other words, we run two models each with the same observed covariates and the tree strucuture of the treatment and control cases will predict counterfactual cases. In the T-learner the ITE is the contrast of the predicted outcomes under M_0 and M_1.

$$\hat{\delta}_i = M_1(X_i) - M_0(X_i) \tag{4.3}$$

Intuition of T-Learners vs S-Learners

A T-learner equivalent to an S-learner that always splits perfectly on the treatment variable at the first decision. This split is guaranteed by using two separate models for the treatment and control observations. This allows the T-learner to bla..

Conceptually, the T-Learner has some opposing properties compared to the S-Learner. While the S-Learner assumes similar structures of the treatment and control groups, the T-Learner assumes different structures and models them separately.

Since the control and treatment groups may exhibit different data structures, we can use different base-learners tailored to each group's characteristics. For example, a linear regression model may be suitable if the control group has linear and additive relationships, especially if interpretability is preferred. If the treatment group shows non-linear relationships, these can be captured using a machine learning model designed to handle these features.

The T-Learner not without fault. When we know there is commonly a zero treatment effect, the T-Learner will perform worse than an S-Learner because it will try to find an effect that is not there. Also, if there are commonalities in the groups, then the T-Learner is less efficient and generalise than the S-Learner as separate models cannot learn from eachother. These properties exist in theory and are corroborated by simulation result.

Although an additional drawback of using the T-Learner is that is can be subject to regularisation bias.

4.4 X-Learners

Finally the X-Learner uses the same first steps as the T-Learning to obtain M_1 and M_0. It imputes the treatment effects for the individuals in the treated group, based on the control-outcome estimator, and the treatment effects for the individuals in the control group, based on the treatment-outcome estimator. Mathematically, we calculate $D_i(1) = Y_i^{obs}(1) - M_0(X_i(1))$ and $D_i(0) = M_1(X_i(0)) - Y_i^{obs}(0)$. In this case, $D_i(1)$ and $D_i(0)$ are the imputed treatment effects for each observation. We then fit two secondary stage learner with the imputed treatment effects as the dependent variables to obtain estimates of $\hat{\tau}_1(x)$ and $\hat{\tau}_0(x)$ the for treatment and control respectively. Finally we use a propensity scores, $e(x)$ and calculate the CATE as a weighted average of the two previous estimates $\text{CATE} = e(x)\hat{\tau}_0(x) + (1 - e(x))\hat{\tau}_1(x)$.

4.5 R-Learners

4.6 Choice of Base-Learner

5 Causal Trees and Forests

Traditional causal inference methods such as regression analysis provide excellent capabilities but also face several limitations. First, many conventional methods assume that treatment effects are homogeneous across the population, which is often unrealistic. In practice, the effect of a treatment can vary significantly across different subgroups or individuals, a concept known as 'treatment effect heterogeneity'. Second, traditional methods struggle with high-dimensional data where the number of covariates is large relative to the sample size. Lastly, the real-world relationships between variables are often non-linear and involve complex interactions that traditional linear models cannot capture adequately. Given these limitations, there is a clear need for non-parametric methods that could flexibly model complex relationships and treatment heterogeneity without imposing strong parametric assumptions. This need motivates the development of machine learning approaches tailored for causal inference such as causal trees and causal forests.

By providing a machine learning approach to handling heterogeneity in treatment effects, causal trees and forests enable researchers and practitioners to draw more targeted and nuanced conclusions from their data, ultimately leading to better-informed decision-making. Specifically of interest is personalised and targeted medicine or policy learning. By modelling treatment effect heterogeneity, there is knowledge of how a particular treatment will impact an individual or a subgroup inside a population. For example, if a medical researcher understands the effect of a medication for a specific subgroup then medical prescription can be personalized. In a policy context, decision-makers could target an intervention or programme towards those who benefit most or those where the benifit exceeds the cost of participation.

Recent research in causal inference has focused on developing methods to estimate treatment effect heterogeneity. This section provides a comprehensive overview of causal trees and forests, including generalized random forest methods. Additionally, these methods are exemplified using the `htetree` and `grf` packages available for the R statistical language.

5.1 Causal Trees: Overview and Intuition

To adapt a Classification and Regression Tree (CART) to a causal context, Athey and Imbens (2016) develop the causal tree (CT). If unfirmilar with CART, please consider reading Section 2.2.1 before proceeding. First, the authors derive a mean square error function allowing trees to identify heterogeneous treatment effects (HTEs). Second, they develop the honesty theorem to ensure desirable asymptotic properties for tree-based estimation of HTEs.

CTs function similarly to CART at a high level by partitioning the data, resulting in purity within child nodes. In a causal context, a better word for purity is *homogeneity*. Instead of splitting for purity (classification tree) or residual sum of squares (regression tree), the CT partitions a given parent node to create nodes where the conditional average treatment effect (CATE) is the most heterogeneous across the resulting child nodes. Athey and Imbens (2016) develop an an expected mean square error formula, EMSE to select splits in the CT. This function introduces a trade-off between maximising the heterogeneity across nodes and minimising the variance of the estimate of the treatment effect within nodes.

There is an intuitive parallel between matching methods and tree methods. Matching methods often match observations based on similarity, usually reduced to a single propensity score as in **?@sec-propensity-scores**. Instead, a causal tree will partition the covariate space and pseudo-match similar observations into child nodes, thus creating neighbourhoods where a comparison between treatment and control cases is robust. Using a tree-based algorithm to define a neighbourhood is advantageous because the distance metric is adaptive. By splitting on covariates that maximise HTEs, CTs identify variables and splits that best explain the heterogeneity. Thus the definition of each neighborhood is adapted to the splits which maximise the heterogeneity. The parallel is important to understand as it motivates the calculation of HTEs and the general philosophy of the tree-based estimation of HTEs.

5.1.1 Honesty Criterion and Estimation of Heterogeneous Treatment Effects

Assuming a tree is already built, the CATE for a given terminal node is denoted $\tau(x;\ell_i)$ and can be estimated simply due to this pseudo-matching property. Recall the calculation of the average treatment effect, denoted τ_{ATE}, from Equation 2.3 is the contrast of treatment and control observations in a sample. A CT estimates $\tau(x;\ell_i)$ separately inside each heterogeneous neighbourhood defined by the tree structure as if it was an independent sample. Hence, across the tree each terminal node, $\tau(x;\ell_i)$ is indexed as $\ell_1,\dots,\ell_n$. Across the tree, $\tau(x;\ell_i)$ is estimated inside each separate terminal node using:

$$\hat{\tau}(x;\ell_i) = \hat{\mu}(x,1;\ell_i) - \hat{\mu}(x,0;\ell_i), \tag{5.1}$$

where $\hat{\mu}$ the estimate of the mean of each treatment group $t \in \{0,1\}$, with attributes x, contained within a given terminal node ℓ_i. In other words, for a given terminal node, $\hat{\tau}(x;\ell_i)$ is the contrast of the mean value between treatment and control observations. The conditional part of the CATE is implied by the structure of the tree preceding the terminal node where a particular $\hat{\tau}(x;\ell_i))$ is evaluated. Each preceding split is interpreted as a *decision rule*, defining the conditions upon which $\hat{\tau}(x;\ell_i))$ is conditioned.

To achieve asymptotically consistent estimates of the treatment effect, Athey and Imbens (2016) propose an "honest" approach, whereby one sample, $\mathcal{S}^{tr}$ is used to construct the partition and another, $\mathcal{S}^{est}$ to estimate treatment effects. In other words, the tree's structure is created using different data than is used to estimate the treatment effect. If the same data is used for training and estimation, extreme values would likely be grouped together during splitting. During estimation, these extreme values would lead to extreme results in some terminal nodes, leading to spurious results created by the chance outliers in a particular sample. Using honesty, the same extreme values will not appear in those terminal nodes, leading to more consistent results. Intuitively, the causal tree must grow the tree structure and estimate the treatment effect. Honest splitting means that half the sample is used for each task. By anticipating honest splitting leading to lower bias, the splitting rule in Equation 5.2 focuses on variance only. Honesty is important to guarantee desirable asymptotic properties and prevent overfitting despite the reduction in sample size for splitting and estimation.

For example, Figure 5.1 is a causal tree trained using $\mathcal{S}^{tr}$ and $\hat{\tau}(x;\ell_i)$ is calculated within the shaded terminal nodes A, B, and C using $\mathcal{S}^{est}$. For node A, $\hat{\tau}(x)$ is evaluated by solving Equation 5.1 with the estimation sample, $\mathcal{S}^{est}$ falling inside A, resulting in an estimate conditioned on $X_1 > 5$. Similarly, for $\mathcal{S}^{est}$ in B, the $\hat{\tau}(x)$ is the difference in mean outcome between treatment and control, conditioned on $X_1 < 5$ and $X_2 > 3.5$.

This approach allows us to observe HTEs between different subgroups which is the motivation for using CTs. It is important to note that subgroups are identified through a data-driven approach rather than prior knowledge. The tree-building process involves splitting the data to discover these groups and estimate their respective $\hat{\tau}(x)$. This property is very advantageous

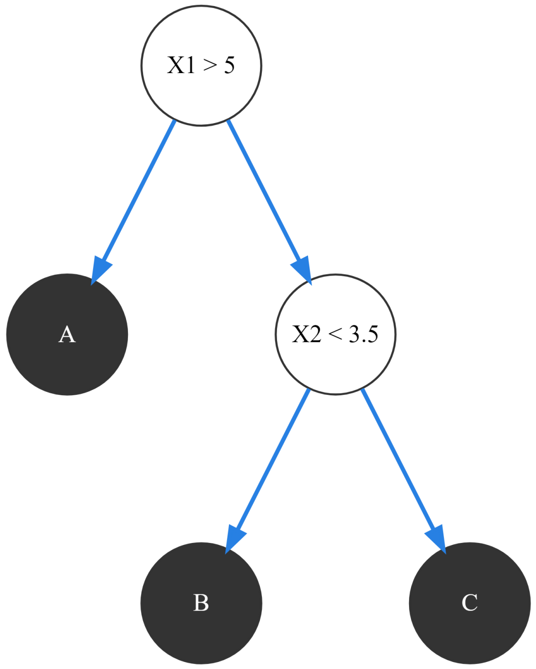


Figure 5.1: Causal Tree Visualisation.

as there are many applications where there is not sufficient prior knowledge of heterogeneity to perform subgroup analysis. In this sense, causal trees are an exploratory research method with the powerful capabilities to find unknown subgroups.

Note 7: Exploratory and Confirmatory Research

The distinction between exploratory and confirmatory research is important to ensure methods like CTs support *good science*. Schwab and Held (2020) note that exploratory research seeks to investigate and understand something without a predefined hypothesis, such as searching for HTEs. Some may describe exploratory research with derogatory terms such as “a fishing expedition” or “data dredging”. On the other hand, confirmatory research tests specific hypotheses derived from theory or prior knowledge. A summary is that exploratory research seeks to find “something”, while confirmatory aims to test a specific “thing”. The two are co-dependent, and both have value in research design.

Problems arise when exploratory and confirmatory research are accidentally confused or deliberately merged. Findings from exploratory research may result in false positives, indicating an effect where none exists. This can happen by chance, poor research design, or from completely spurious relationships. The false positive problem worsens as the amount of "exploration" increases, such as assessing more subgroups or using additional analysis methods. Additionally, a researcher may only present the interesting or convenient results leading to "cherry picking" of results. Additionally, the researcher may find the results and then specify the hypothesis as if it was previously known. This can be analogised as throwing darts then drawing the dartboard around the landing spot.
The two work best together in a research cycle where exploratory research generates hypothesis and confirmatory results tests them. Methods such as CTs can generate hypotheses, and confirmatory research can consider research design to test the hypothesis optimally as a part of a pre-registered study plan.

5.1.2 Technical Detail and Splitting Criteria

Typically a mean square error (MSE) function takes a form similar to $\frac{1}{n}\sum(y_i - \hat{y_i})^2$ where y_i is the true value, $\hat{y_i}$ is the estimated value, and n is the number of observations. Intuitively, the typical mean square error says to make the predicted values the closest to the actual values. The fundamental problem discussed in Note 1 means the MSE is not appropriate as we never observe true value of the individual treatment effect (ITE), denoted τ_i.

Recall from the introduction that a CT splits to maximise heterogeneity across nodes while minimising variance of the treatment effect within nodes. To maximise heterogeneity,the CT will maximise the variance of the predictions Equation 5.1 across all leaves. More variance across terminal nodes implies that the CT has found more heterogeneity of the treatment effect across the neighbourhoods identified by pseudo-matching. At the same time splits should be made so that the estimates of the treatment effect are robust and so the the splitting rule includes variance penalisation as well.

Putting these elements together, Athey and Imbens (2016) develop an expected mean square error function suitable for an unobserved τ_i:

$$\begin{aligned} -\widehat{\text{EMSE}}_\tau\left(\mathcal{S}^{tr}, N^{est}, \Pi\right) \equiv \alpha \frac{1}{N^{tr}} \sum_{\ell \in \Pi} \hat{\tau}^2\left(x; \mathcal{S}^{tr}, \Pi\right) \\ -(1-\alpha)\left(\frac{1}{N^{tr}}+\frac{1}{N^{est}}\right) \cdot \sum_{\ell \in \Pi}\left(\frac{\mathcal{S}^2_{\mathcal{S}^{tr}_{\text{treat}}}(\ell)}{p}+\frac{\mathcal{S}^2_{\mathcal{S}^{tr}_{\text{control}}}(\ell)}{1-p}\right). \end{aligned} \tag{5.2}$$

Let, - $\frac{1}{N^{\text{tr}}} \sum_{\ell \in \Pi} \hat{\tau}^2\left(x; \mathcal{S}^{\text{tr}}, \Pi\right)$ be the variance of the treatment effect across nodes in the training sample, $\mathcal{S}^{\text{tr}}$. This expression extends Equation 5.1 across all terminal leaves contained within the tree such that $\ell_1, \ell_2 \ldots \ell_n \in \Pi$.

- N^{tr} and N^{est} be the number of observations in the training and estimation samples.
- $\mathcal{S}^2(\ell)$ be the variance of the estimate of the treatment effect inside terminal node ℓ.
- p be the treatment share expressed as $N^{\text{tr}}/N^{\text{est}}$.
- α be a scalar controlling the relative contributions of each part of the equation.

Note 8: Relating back to Intuition

The intuition behind this complex error function is key to understanding causal trees. The first term is the estimated variance of the treatment effect across leaves by definition. Similar to the general definition of variance, it is the sum of the treatment effect across terminal nodes divided by the number of training observations. Each terminal node contributes to this term and an increase in the variance across terminal nodes indicates more heterogeneity of the treatment effect (it is more different across terminal nodes.). The second term is a measure of the variance of the treatment effects within each node. The logic of the equation means that an increase in treatment effect heterogeneity between terminal nodes will reduce the (negative) of the EMSE and an increase in the uncertainty of the node estimates will increase the EMSE. Thus we can say that the EMSE equation will reward creating heterogeneous partitions and heterogeneous terminal nodes while also obtaining appropriate estimates of the treatment effects within each terminal node.

Equation 5.2 controls the relative strength of the two terms using the α parameter. A high value of α means that the variance of the treatment effect is less important. We expect that a higher α value will increase the depth of the trees as splits will ignore the variance of the estimated treatment effect. Contrastingly, $\alpha = 0$ means that a tree will make no splits as this will make the variance of the treatment effect the lowest. If $\alpha = 1$, then trees are grown very deep and will continue to split until a defined number of minimum observations in each node is reached.

Interestingly, no pruning is required after the CT model is fit as the complexity of trees is implicitly included in Equation 5.2. If a tree gets too complex, then the variance in the terminal node estimates will increase on unseen data in S^{est}. The second part of Equation 5.2 punishes the model for an increase in variance in terminal nodes. Thus, the splits that lead to a relatively large variance of the $\hat{\tau}(x; \ell_i)$ will not be made. This variance penalisation also implies that small leaves will be penalised as small leaves lead to higher variance.

5.1.3 Cross-Validation

Cross validation does play an important role in a causal tree but not to the same extent as in CART. Cross validation is made complex as τ_i is unknown and there is not a test/train split present as in many other machine learning applications.

A CT requires honest cross validation so that the tree can discover the structure of the tree using unseen data. Noting that to make a split, the treatment effect must be evaluated in the potential child nodes. There is a cross-validation subset within $\mathcal{S}^{tr}$ that is used to evaluate Equation 5.2. In other words, when training the tree to discover its structure, we use the cross validation set, $\mathcal{S}^{cv}$, to evaluate splits. When doing this honest cross validation, the $\widehat{\text{EMSE}}$ is the same as in Equation 5.2 but uses $\mathcal{S}^{tr,cv}$ instead of $\mathcal{S}^{tr}$. Note that $\mathcal{S}^{est}$ is used after the tree is trained for the final estimation inside the terminal nodes.

5.1.4 Implimentation inside R: `htetree`

A causal tree can be implemented inside R using the `htetree` package which provides a large library of functions for estimating heterogeneous treatment effects with tree-based machine learning algorithms as well as visualisation.[1] This package uses `rpart` which is a common implementation of CART in R.

My exposition of the CT method does not explore the results of the analysis in depth as this is explored further in the (**sec?**)-. I perform a basic fit which demonstrates the implementation followed by a simple visualisation which reinforces the intuition of a causal tree.

```
set.seed(88)
library(caret)                                                        (1)
nsw_split <- createDataPartition(nsw_data$data_id, p=0.7,
                                 list = FALSE)
nsw_train <- nsw_data[nsw_split,]
```

[1]It appears to me that the `htetree` package has a more comprehensive implimentation compared to the original `causalTree` package created by Susan Athey which is hosted on Github.

```r
nsw_est <- nsw_data[-nsw_split,]


library(htetree)                                                              ②
library(rpart)
nsw_causalTree <- honest.causalTree(re78 ~ age + educ +                       ③
                                    black + hisp + degree + marr,
                                    data = nsw_train,
                                    treatment = nsw_train$treat,
                                    est_data = nsw_est,
                                    est_treatment = nsw_est$treat,
                                    split.Rule = "CT", split.Honest = T,
                                    HonestSampleSize = nrow(nsw_est),
                                    cv.option = "CT", split.alpha = 0.5,
                                    minsize = 5)                              ④
```

① The data is split into the training and estimation set as required to fit an honest tree. This is done using the `createDataPartition` function in the `caret` package. Following a common rule-of-thumb, 70% of the data is used for training and 30% is used for estimation.

② The `htetree` and `rpart` packages are loaded. Both are required to fit an `honest.causalTree` as `htetree` uses `rpart` to fit the model.

③ The parameters of the `honest.causalTree` function are specified. The process of specifying some of these objects is cumbersome.

④ I have specified `minsize = 5` meaning that each node must contain 5 or more cases of both treated and control observations. *This is set for ease of visualisation not advantageous model fit.*

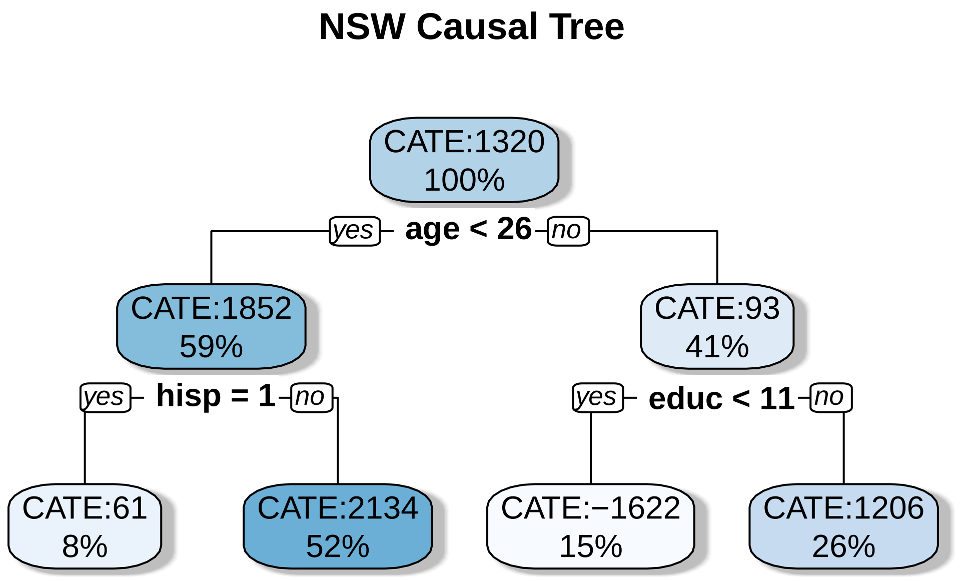


Figure 5.2: This visualisation shows the splits selected by the Causal Tree. Darker shaded nodes correspond to a higher estimate of the CATE. Percentages indicate the percent of observations falling into a given node.

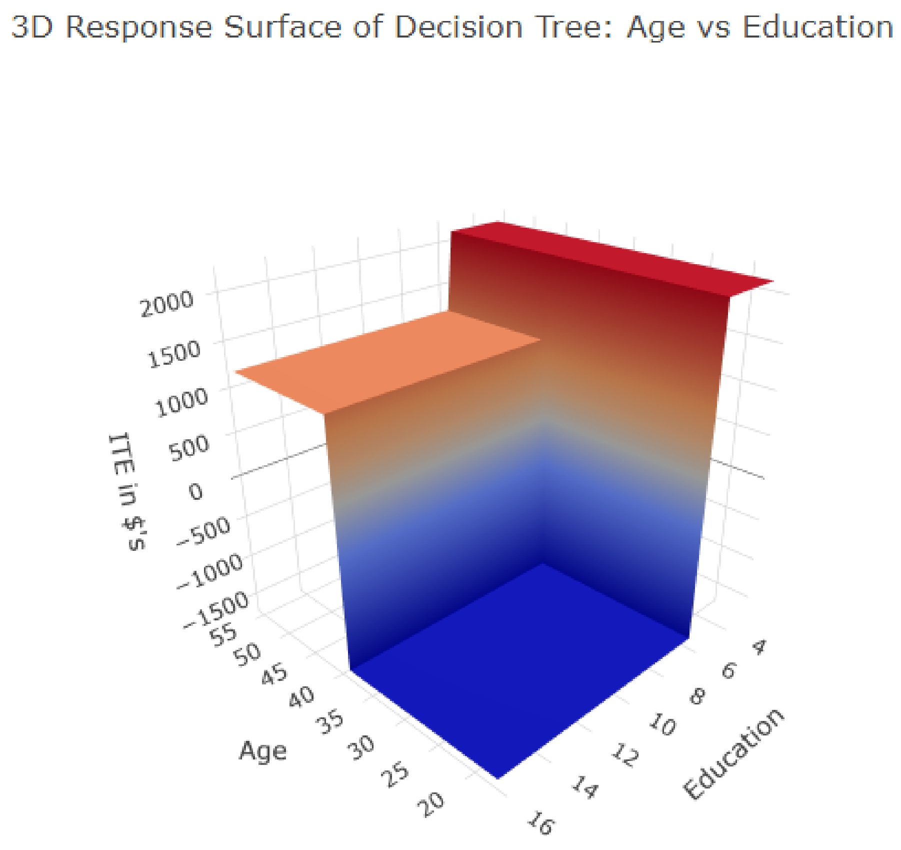


Figure 5.3: The interaction between age and education is shown in the response surface of the causal tree. The visualisation is interactive when viewing as a .html file type

To produce Figure 5.3, all other variables have been held constant at their mean values. This is not a problem as the splits involving age and education do not interact with any other covariates.

The CT algorithm has identified heterogeneous subgroups and has calculated $\hat{\tau}(x)$ for heterogeneous subgroups. However, CT have some similar drawbacks to regular CART models. First, this is a *greedy* algorithm meaning that splits at earlier points are made without consideration of later splits. Secondly, the algorithm is likely to lead to quite sharp decision boundaries as visualised in Figure 5.3. In this example, an individual is younger than 26, an increase in education from 10 to 11 results in a completely different $\hat{\tau}(x)$. This is unrealistic and smoother smooth decision boundaries should better replicate the real world.

5.1.5 Towards a Causal Forest

The next natural development upon causal trees (CT) is a causal forest (CF). Theoretically, a CF is preferable to a CT because ensemble learning methods such as random forest (RF) typically result in superior performance. An ensemble of CTs should smooth sharp decision boundaries, leading to much better estimates of $\tau(x)$. Consider $\tau(x)$ to be a continuous CATE function in the context of a causal forest.

Wager and Athey (2018) present a simple implementation of a CF that averages the predictions from many individual CTs, each optimising the EMSE from Equation 5.2. Following the conventional philosophy of random forest as in Brieman (2001), averaging the predictions across diverse trees should lead to improved predictions and smooth decision boundaries. The final prediction from this implementation of a CF is $\hat{\tau}(x) = \frac{\sum_{b=1}^{B} \hat{\tau}_b(x)}{B}$ where B is the number of trees and $\hat{\tau}_b(x)$ is the CATE calculated by each base tree.

Random forests achieve diverse trees by fitting each tree on a bootstrapped sample of the data. In other words, each tree is grown with different observations, leading to more tree diversity. Fitting individual trees with bootstrapped observations is often referred to as *bagging* or *bootstrap aggregation*, which was introduced by Leo Breiman (1996). To further encourage tree diversity, several randomly selected variables are considered at each split to reduce the correlation between trees further. Splitting on different variables at each split prevents strong predictors from always being the initial variables selected, allowing trees to model interactions between other variables.

A CF cannot directly identify subgroups as the overall prediction is an individual treatment effect. The only reason CTs can identify subgroups is because sharp boundaries define heterogeneous groups. In a CT, averaging of predictions should smooth boundaries, thus making subgroups not directly interpretable.

A key contribution of Wager and Athey (2018) is the asymptotic normality of random forests, enabling valid statistical inference. The predictions made by a causal forest are asymptotically normal and unbiased, provided unconfoundedness and common support assumptions are met. The variance predictions from a CF can be estimated using a jackknife procedure developed by Wager, Hastie, and Efron (2014), and the estimation of confidence intervals is possible.

While this implementation of a CF is appealing, there are two downsides. First, the predictions from a CF may be poor as individual CTs may be biased even when using sample splitting due to low sample size or poor tuning of the CT algorithm. Additionally, errors from tree predictions of $\hat{\tau}(x)$ may be so large that the average across tree predictions is still a poor estimate of the treatment effect. Secondly, computing splits for each tree is very time-consuming, given that Equation 5.2 is a highly complex function. These weaknesses and a desire for a generalised approach led Athey, Tibshirani, and Wager (2019) to develop the Generalized Random Forest.

5.2 Generalized Random Forests: Overview and Intuition

Causal forests belong to a family of forests called *Generalized Random Forests* (GRF), introduced by Athey, Tibshirani, and Wager (2019). As suggested in the name, these forests are a generalization of a random forest (RF) suited to diverse applications such as survival analysis or instrumental variable methods. At first, I first provide a simple overview of GRF and omit the details of implementation to simplify my explanation. Then, I introduce GRF on a technical level in Section 5.2.1. However, the following explanation does not replace studying the original paper Athey, Tibshirani, and Wager (2019).

Note 9: Different Implimentations of a Causal Forest

To be very clear, the implementation of a CF in a GRF framework is different than a simple average of trees discussed in Section 5.1.5. More specifically, the implementation of causal forests in Wager and Athey (2018) and Athey, Tibshirani, and Wager (2019) are different. The implementation in the `grf` software follows Athey, Tibshirani, and Wager (2019). An "honest causal forest" using averages can be implemented with the `htetree` package. The distinction between the implementations is commonly mistaken in the literature and care must be taken when discussing these methods as the literature continues to develop. Notably, Jawadekar et al. (2023) make the mistake of conflating the two implementations in their paper which aimed to clarify the implimentation of a causal forest.

Recall from Section 5.1 that a causal tree can be analogised to a neighbourhood pseudo-matching method where observations in the same terminal node are "matched" together. Athey, Tibshirani, and Wager (2019) take the neighbourhood analogy further and consider GRF to be an "adaptive locally weighted estimator that first uses a forest to calculate a weighted set of neighbours for each test point, then solve a plug-in version of a [local estimating equation]". Conceptually, a local estimating equation is like an equation to calculate the treatment effect but optimised locally allowing it to represent HTEs. Different specifications of the GRF framework are implemented by changing the form of the estimating equation.

The *forest* part of a GRF develops a weighting scheme that defines how similar observations are to each other. A single tree can create groups of observations inside nodes in a neighbourhood characterised by splits leading to the terminal node. For example, a neighbourhood in Figure 5.1 comprises observations falling into each terminal node. Instead of a binary "in the neighbourhood" or "outside the neighbourhood", a forest provides a scale of closeness defined by weights. The weights for an arbitrary observation, x, capture the frequency of when that observation falls into the same terminal node as other observations in each tree inside the forest.

To estimate these weights, A GRF uses a modified random forest called GradientForest that incorporates a gradient descent process at each split to maximise the observations' total "influence" upon the parameter estimates. Each tree is grown to be "deep" using this influence-based splitting rule and a gradient descent sequence. Notably, EMSE from Equation 5.2 does not feature in the GRF algorithm as it is difficult to optimise and does fit the framework of a GRF.

A GRF uses these similarity weights to evaluate a local estimating equation that captures heterogeneity of a treatment effect. Framing random forests as a method to estimate similarity weights contradicts conventional random forests. Typically, a random forest grows many trees and then averages the predictions so that the average of each tree is a much better model than any individual tree. In GRF, each tree is used to compute weights to solve a locally-weighted heterogeneous equation.

5.2.1 Technical Explanation of GRF

Keeping neighbourhood finding in mind, consider that there exists some heterogeneous estimating equation in the form of a localised generalized method of moments (LGMM) estimator, as in Lewbel (2007). This LGMM equation is of the general form:

$$\mathbb{E}\left[\psi_{\theta(x),\nu(x)} \mid X = x\right] = 0, \tag{5.3}$$

where $\psi_{\theta(x),\nu(x)}$ is a moment condition that I refer to as the score function. Specifically, $\theta(x)$ is an estimator of some desirable quantity such as a conditional mean, quantile or average partial effect and ν is a nuisance parameter that exists in some specifications of a GRF. Equation 5.3 must hold for the local parameter estimates $\theta(x)$ to be unbiased and consistent.

Note 10: A Quick Primer on Method of Moment Estimation

The Method of Moments (MoM) estimation is a method to estimate parameters in the form of moments. The method consists of equating theoretical moments to sample calculations (sometimes called an empirical moment). For example, the first moment, $E[X]$ is the mean with the sample calculation $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$. Generalized Method of Moments (GMM) extends MoM by incorporating multiple moment conditions and a weighting matrix, allowing for more efficient and robust parameter estimation and when there are more moment conditions than parameters. Local Generalized Method of Moments (LGMM) further extends GMM by applying moment conditions locally, using a weighting matrix to focus on localities of data. This approach can capture heterogeneity and provide parameter estimates that vary smoothly across the sample, making it useful for modelling heterogeneous treatment effects.

The solution to the LGMM estimator is found at some optimisation value. In a CF, the desirable quantity is the CATE. A GRF solves an LGMM estimator using weights, $\alpha_i(x)$ provided by a GradientForest to obtain a smooth response surface where the global variation in the surface represents treatment effect heterogeneity. The solution to the local equation is

$$\left(\hat{\theta}(x), \hat{\nu}(x)\right) \in \arg\min_{\theta,\nu} \left\| \sum_{i=1}^{n} \alpha_i(x) \psi_{\theta,\nu}(O_i) \right\|_2 . \tag{5.4}$$

A GRF can model any quantity of interest identified by a local moment condition. To specify the type of GRF, there must be an empirical version of the local moment equation to optimise. For example, in a conventional regression random forest, this local estimation equation is simply the residual sum of squares (RSS), and so a GRF implementation of a regression random forest is practically identical to the standard implementation as in `randomForest`.

As the success of this method relies on estimating weights that represent heterogeneity, the forest used to calculate these weights is not a typical RF. As mentioned above, a GradientForest is optimised to create heterogeneous splits while also being computationally efficient. Being a *generalized* method, this splitting rule can take many forms but should encourage heterogeneity so the resulting weights can represent the heterogeneity of the treatment effect.

To make computation possible, Athey, Tibshirani, and Wager (2019) implement a gradient descent process where the gradient of $\psi_{\hat{\theta}_P,\hat{\nu}_P}$ is computed once at each split. Let $\hat{\theta}_P$ and $\hat{\nu}_P$ be estimates of θ and ν in parent node, P. Each tree separately performs a gradient descent procedure similar to a Gradient Boosting Machine (GBM) introduced by Friedman (2001). The key difference is that a GBM computes the gradient once per tree and sequentially descends a loss function across the whole ensemble. In a GradientForest, each split is an iteration towards the minimum of the splitting rule, and each tree separately performs gradient descent. In other words, a GBM descends from tree to tree while a GRF descends from split to split within each tree.

The GRF gradient descent process iteratively applies two steps. First is creating what Athey, Tibshirani, and Wager (2019) describe as pseudo-outcomes in a "relabelling step". Inside each parent node, P, the LGMM estimator in Equation 5.4 is solved without weights to estimate the desirable quantity $\widehat{\theta}(x)$. [2] Also, the Hessian matrix of $\psi_{\hat{\theta}_P,\hat{\nu}_P}$ is computed once inside a parent node. This matrix A_P is the Hessian matrix $\nabla\psi_{\hat{\theta}_P,\hat{\nu}_P}$ evaluated in P, normalised over the number of observations inside the parent node:

$$A_P = \frac{1}{|\{i : x_i \in P\}|} \sum_{\{i:x_i \in P\}} \nabla\psi_{\hat{\theta}_P,\hat{\nu}_P}(O_i). \tag{5.5}$$

To calculate pseudo-outcomes, let $-\xi^\top$ be a operator that "picks out" the vector of $\widehat{\theta}(x)$ from a the matrix. Pseudo-outcomes are a combination of the score function and the Hessian matrix of the score function evaluated at the current $\theta(x)$ estimate using observations within the parent node:

$$\rho_i = -\xi^\top A_P^{-1}\psi_{\hat{\theta}_P,\hat{\nu}_P}(O_i). \tag{5.6}$$

Pseudo-outcomes are analogous to pseudo-residuals in a GBM. Additionally, pseudo-outcomes are a measure of influence that measures how much the omission of observation i changes the parameter estimate.[3]

[2]The estimation of $\widehat{\theta}(x)$ in P, cannot use weights, $\alpha_i(x)$ as these have not been estimated yet.

[3]Note this step aligns with the general form of the influence function. Specifically from Weisberg and Cook (1982): $I(z) = -H_{\hat{\theta}}^{-1}\nabla_\theta\Psi(z,\widehat{\theta})$. $I(z)$ is the influence function for the point z. $H_{\hat{\theta}}^{-1}$ is the inverse of the Hessian matrix of the log-likelihood (or another objective function). $\nabla_\theta\Psi(z,\widehat{\theta})$ is the gradient of the function Ψ.

Viewing these pseudo-outcomes as influence measures, splits that maximise the level of influence in the child nodes should approximately correspond to splitting directly for the desirable quantity. Being a generalized method, the intuitive representation of ρ_i depends on the nature of the score function. If $\psi_{\hat{\theta}_P,\hat{\nu}_P}(O_i)$ is the RSS, then splitting to maximise ρ_i is approximately equivalent to splitting to minimise RSS. If $\psi_{\hat{\theta}_P,\hat{\nu}_P}(O_i)$ is some equation to represent heterogeneity, then maximising ρ_i is approximately equivalent to splitting directly for heterogeneity.

Splitting to maximise the total influence across all observations is computationally efficient compared to optimising a complex function such as Equation 5.2. Specifically, the calculation of the split is normalised over the number of observations in each child node C_j. Let $\tilde{\Delta}(C_1, C_2)$ be a measure of influence in the child nodes:

$$\tilde{\Delta}(C_1, C_2) = \sum_{j=1}^{2} \frac{1}{|\{i : x_i \in C_j\}|} \left(\sum_{\{i : x_i \in C_j\}} \rho_i \right)^2 . \tag{5.7}$$

The GRF algorithm grows the first of many diverse trees by greedily splitting data. As in Section 5.1.5, each tree uses a bootstrapped sample of data, and each split uses a certain number of randomly selected variables.

Let the set of B trees inside the forest be indexed as $b = 1, \dots, B$ and let $L_b(x)$ be the set of observations that fall into the same terminal node as arbitrary value x. Weights $\alpha_i(x)$ capture the frequency with which the ith training example falls into the same terminal node as x such that

$$\alpha_i(x) = \frac{1}{B} \sum_{b=1}^{B} \frac{\mathbf{1}\{x_i \in L_b(x)\}}{|L_b(x)|}, \tag{5.8}$$

where **1** is an indicator function equal to 1 if x_i is in the same node as arbitrary point x and $|L_b(x)|$ is the number of observations in the node where x falls. For every observation where the score function is evaluated, a corresponding weight matrix defines how similar that point is to all other points in the data. As previously mentioned, these weights are used to solve Equation 5.4 to obtain $\hat{\theta}(x)$.

5.2.2 Causal Forests in the General Framework

Recall from Section 2.1.9 that a model describes the linear relationship between the treatment, covariates, and outcome, as in Equation 2.10. Instead, suppose a non-linear relationship exists between covariates and outcomes and heterogeneous treatment effects. This model is stated as

$$Y = \tau(x)T + f(x) + \epsilon, \tag{5.9}$$

where $\tau(x)$ is the CATE and $f(x)$ is a function describing the relationship between covariates and the outcome. Recall from Chapter 3 that the propensity score reduces the dimension of $f(x)$ to a single propensity score, $e(x)$. However, a propensity score approach does not allow for HTEs. In the framework of a GRF, $\tau(x)$ is solved using weights defined by the GradientForest procedure to estimate Equation 5.3.

Estimating Equation 5.9 seems simple but is theoretically problematic. Conceptually, a model must "understand" some elements of the data to identify HTEs. First, it must model the effect of the covariates on the outcome and potential interactions between covariates so treatment effects are isolated. Second, it must model the relationships between covariates and the treatment as these relationships drive heterogeneity. Third, it should model the conditional effect of the covariates on the probability of being treated so estimates are not subject to confounding. Finally, it must also estimate weights, $\alpha_i(x)$, that result in good estimates of the local equation. Using a single model risks "overwhelming" the model, leading to sub-optimal weights and poor treatment effect estimation. To simplify the estimation, a causal forest uses an approach called "local centering" or orthogonalisation, introduced by Robinson (1988). A Robinson transformation will "regress out" the conditional effects of the covariates on the outcome and treatment indicator.

Let $\mu(x)$ be the conditional outcome such that

$$\mu(x) = E[Y|X = x] = f(x) + \tau(x)e(x). \tag{5.10}$$

Subtracting the conditional outcome, $\mu(x)$, orthogonalises Equation 5.9:

$$\begin{aligned}
Y &= \tau(x)T + f(x) + \epsilon \\
Y - \mu(x) &= \tau(x)T + f(x) - m(x) + \epsilon \\
Y - \mu(x) &= \tau(x)T + f(x) - (f(x) + \tau(x)e(x)) + \epsilon \\
Y - \mu(x) &= \tau(x)(T - e(x)) + \epsilon
\end{aligned}$$

In the `grf` package, the `causal_forest` function estimates $\hat{e}^{(-i)}(x)$ and $\hat{m}^{(-i)}(x)$ using separate regression forests and then plugs the estimates into in the orthogonalised function:

$$Y - \hat{m}^{(-i)}(x) = \tau(x)(T - \hat{e}^{(-i)}(x)) + \epsilon. \tag{5.11}$$

The superscript $^{(-i)}$ indicates that the estimate for i is made on a sample not including i. Nie and Wager (2021) develop a loss function for treatment effect estimation of this general form. Derived from Equation 5.11, the score function is

$$\psi_\tau = (Y - \hat{m}^{(-i)}(x) - \tau(T - \hat{e}^{(-i)}(x)))(T - \hat{e}^{(-i)}(x)), \tag{5.12}$$

with the moment condition defined by:

$$E[Y - \hat{m}^{(-i)}(x) - \tau(T - \hat{e}^{(-i)}(x)))(T - \hat{e}^{(-i)}(x)] = 0. \tag{5.13}$$

Robinson (1988) shows us that an orthogonalisation approach is theoretically advantageous as noisy estimates of $\hat{m}^{(-i)}(x)$ and $\hat{e}^{(-i)}(x)$ will still lead to good estimates of $\tau(x)$. Particularly, $\hat{m}^{(-i)}(x)$ and $\hat{e}^{(-i)}(x)$ can converge at a slower rate but $\hat{\tau}(x)$ remains robust. Additionally, by specifying $e(x)$, the CF should be robust to confounding provided that Equation 2.7 is satisfied.

Finally, the final definition of a causal forest is the optimisation of

$$\hat{\tau}(x) \in \arg\min_{\tau} \left\| \sum_{i=1}^{n} \alpha_i(x)(Y - \hat{m}^{(-i)}(x) - \tau(T - \hat{e}^{(-i)}(x)))(T - \hat{e}^{(-i)}(x)) \right\|_2, \tag{5.14}$$

using weights, $\alpha_i(x)$, obtained through a GradientForest.

5.2.3 Other Generalised Random Forests Algorithems Inside `grf`

The `grf` package provides a range of random forest implementations including `causal_forest`, `survival_forest`, `instrument_forest`, `quantile_forest`, and `probability_forest`. There is notable documentation provided on the package Github.

5.3 Application: National Supported Work Data (DONT READ, some things have run into errors which will hopefully be a simple fix. they previously worked fine.)

While the NSW dataset has often been applied in the context of propensity score methods in textbook examples. Seldom has treatment effect heterogeneity been explored. The Causal Tree earlier suggested that age and education may be significant indicators of the success of the programme for individuals. Using the `grf` package, the model is fit using all covariates including income in earlier years.

```
library('grf')
library(dplyr)
covars <- nsw_data %>% select(age, educ, black, hisp, marr, degree, re74)

nsw_cf_model <- causal_forest(
  X = covars,                                                      ①
  Y = as.vector(nsw_data$re78),
  W = as.vector(nsw_data$treat),
  tune.parameters = "all",
  seed = 88)                                                       ②

  nsw_ite_predict <-predict(nsw_cf_model, estimate.variance = TRUE)  ③
  nsw_ite <- nsw_ite_predict$predictions                           ④
```

```r
install.packages('grf')
library('grf')

nsw_cf_model <- causal_forest(
  X = nsw_covariates,                                                    ①
  Y = nsw_dependant,
  W = nsw_treatment,
  tune.parameters = "all")                                               ②

  nsw_ite_predict <-predict(nsw_cf_model, estimate.variance = TRUE)      ③
  nsw_ite <- nsw_ite_predict$predictions                                 ④
```

① The observed covariates, dependant variable, and treatment variable are specified as `X`, `Y`, and `W` respectively. Note that these will be in vector form such as `as.vector()`.

② The `causal_forest()` function will tune for `sample.fraction`, `mtry`, `min.node.size`, `honesty.fraction`, `honesty.prune.leaves`, `alpha`, and `imbalance.penalty`. These can be manually specified but that is beyond the scope of this book. Further information can be obtained inside R using `?causal_forest`.

③ Predictions are made from the `nsw_cf_model` object and `estimated.variance` is set `TRUE` because reason.

④ The ITE are extracted from the `nsw_ite_predict` object.

Using the the Causal Forest model, `nsw_cf_model`, a response surface can be displayed as in Figure 5.3. As before, this is estimated holding all other variables at their mean. This visualisation is somewhat dubious as the various trees inside the forest will have split on other variables as well as education and age so holding them constant somewhat undermines the model structure. However, for the purposes of demonstration, this visualisation is helpful regardless.

```r
library(plotly)
library(grf)
grid <- expand.grid(age = age_seq,educ = educ_seq)

mean_values <- colMeans(nsw_data, na.rm = TRUE)
other_vars <- names(nsw_data)[!(names(nsw_data) %in%  c("age", "educ",
                                   "response","ite", "re78","re75", "data_id","treat"))]

for (var in other_vars) {
grid[[var]] <- mean_values[var]
}

grid$predictions <- predict(nsw_cf_model, newdata = grid)

response_matrix <- matrix(as.numeric(unlist(grid$predictions)),
                          nrow = length(educ_seq), ncol = length(age_seq))

fig <- plot_ly(
x = ~age_seq,
y = ~educ_seq,
z = ~response_matrix,
type = 'surface',
colorscale = list(
c(0.25, 'rgb(0, 255, 255)')
),
showscale = FALSE
)

camera <- list(
eye = list(x=-1.7, y=-1.7, z=1.7),
center = list(x = 0, y = 0, z = 0),
up = list(x = 0, y = 0, z = 1)
)



fig <- fig %>% layout(
scene = list(
xaxis = list(title = 'Age'),
yaxis = list(title = 'Education'),
zaxis = list(title = "ITE in $'s"),
camera = camera
),
title = '3D Response Surface of Causal Forest: Age vs Education'
)
fig
```

3D Response Surface of Causal Forest: Age vs Education

Figure 5.4: rewrite: The interaction between age and education is shown in the response surface of the causal Forest The visualisation is interactive when viewing as an html web file.

While the response surface is still sharp in some places, there is a visible gradient showing the increase

Additionally, using the causal forest, the estimates of the average treatment effect seem more likely. The effect is predicted to be positive for all individuals when holding all other individuals at their mean values. However, the general relationship and heterogeneity in the data is the same. The Causal Forest finds that increasing age and increasing education results in a higher individual treatment effect.

```
# library(ggplot2)
# library(ggthemes)
# library(scales)
# ggplot(nsw_data) +
#   aes(x = as.factor(degree), y = ite) +
#   geom_boxplot()
```

```
# ggplot(nsw_data) +
#   aes(x = as.factor(marr), y = ite) +
#   geom_boxplot()
```

```
# ggplot(nsw_data) +
#   aes(x = as.factor(black), y = ite) +
#   geom_boxplot()
```

```
# ggplot(nsw_data) +
#   aes(x = as.factor(hisp), y = ite) +
#   geom_boxplot()
```

```r
library(ggplot2)

library(ggthemes)

library(scales)

ggplot(nsw_data, aes(x = educ, y = nsw_ite)) +

  geom_point(size=2) +

  labs(

    title = "Visualising Treatment Effect Heterogeneity",

    subtitle = "Years of Education vs. Estimated Individual Treatment Effect",

    y = "Individual Treatment Effect ($ Increase in Earnings in 1978)",

    x = "Years of Education") +

  geom_rangeframe() +

  theme_tufte(base_size = 14)+

  theme(

    plot.title = element_text(size = 20),

    plot.subtitle = element_text(size = 15),

    legend.position = "bottom") +

  scale_x_continuous(breaks = pretty_breaks(n = 15)) +

  scale_y_continuous(breaks = pretty_breaks(n = 10))
```

```r
library(ggplot2)

library(ggthemes)

library(scales)

ggplot(nsw_data, aes(x = age, y = nsw_ite, color = as.factor(degree))) +

  geom_point(size=2) +

  labs(

    title = "Visualising Treatment Effect Heterogeneity",
    subtitle = "Age of Participant vs. Estimated Individual Treatment Effect",
    y = "Individual Treatment Effect ($ Increase in Earnings in 1978)",
    x = "Age (Years)") +
  geom_rangeframe() +
  theme_tufte(base_size = 14)+
  theme(
    plot.title = element_text(size = 20),
    plot.subtitle = element_text(size = 15),
    legend.position = "bottom") +
  scale_x_continuous(breaks = pretty_breaks(n = 15)) +
  scale_y_continuous(breaks = pretty_breaks(n = 10)) +
  scale_color_manual("Bachelor Degree Completion:",values = c("1" = "red",
                                                              "0" = "black" ),
                     labels = c("1" = "Degree","0" = "No Degree"))
```

This result makes sense heuristically for a couple of reasons. Perhaps older individuals are more likely to appreciate stable employment opportunities, leading to more engagement with the programme. (**lalonde?**) notes that older and more educated poeple completed the programme at a higher rate demonstrating their commitment. Perhaps the training programme taught skills this demographic was disproportionately lacking. Perhaps more educated individuals experience higher treatment effects due to better job readiness and adaptability to new training. Perhaps different demographics suffered from unemployment for different reasons leading to heterogeneous results.

I should remind readers of an important caveat, these results are found through exploratory research and are not confirmatory. However, they are an indication that there is substantial heterogeneity present which further research can confirm the results of. This further research likely requires a different research design to isolate the heterogeneity caused by age and education. This result could be entirely spurious or due to an unobserved confounders.

Perhaps the older demographics were motivated workers but unemployed/underemployed because of structural changes in the economy such as the general decline in manufacturing jobs in the late 1970's in the United States. Meanwhile younger applications may have personal reasons for unemployment/underemployment meaning they were less driven or motivated to to obtain higher earning jobs after the programme. While this speculation is unlikely to be the cause of this heterogeneity, it does reinforce the idea that exploratory research should not be confused wth confirmatory research. There are many hypothetical confounding relationships that exist which can be speculated upon but the key idea is that the research design does not control for some of these reasons and so we cannot conclude from the exploratory finding that there is a confirmation increased age or education leading to higher treatment effects in work programmes.

5.4 Code Provided for PDF Output

```r
load(file = "my_environment.RData")
# This code creates the rpart plot to visualise Figure 1plageholder
library(rpart.plot)
rpart.plot(nsw_causalTree, yesno = 2, shadow.col = "grey",
           prefix = "CATE:", main = "NSW Causal Tree")
# This code creates the rpart plot to visualise Figure placeholder
library(plotly)

age_seq <- seq(min(nsw_data$age), max(nsw_data$age), length.out = 50)
```

```r
educ_seq <- seq(min(nsw_data$educ), max(nsw_data$educ), length.out = 50)
grid <- expand.grid(age = age_seq,educ = educ_seq)

mean_values <- colMeans(nsw_data, na.rm = TRUE)
other_vars <- names(nsw_data)[!(names(nsw_data) %in%
                                  c("age", "educ", "response"))]

for (var in other_vars) {
grid[[var]] <- mean_values[var]}
grid$response <- predict(nsw_causalTree, newdata = grid)
response_matrix <- matrix(grid$response, nrow = length(educ_seq),
                          ncol = length(age_seq))

fig <- plot_ly(
  x = ~age_seq,
  y = ~educ_seq,
  z = ~response_matrix,
  type = 'surface',
  colorscale = list(
    c(0, 'rgb(0, 0, 255)'), # Dark blue
    c(0.25, 'rgb(0, 255, 255)'),# Cyan
    c(0.5, 'rgb(0, 255, 0)')), # Green
  showscale = FALSE)

camera <- list(
  eye = list(x=-1.6, y=1.6, z=1.1),
  center = list(x = 0, y = 0, z = 0),
  up = list(x = 0, y = 0, z = 1))

fig <- fig %>% layout(
  scene = list(
    xaxis = list(title = 'Age'),
    yaxis = list(title = 'Education'),
    zaxis = list(title = "ITE in $'s"),
    camera = camera),
  title = '3D Response Surface of Decision Tree: Age vs Education')

fig
library('grf')
library(dplyr)
covars <- nsw_data %>% select(age, educ, black, hisp, marr, degree, re74)

nsw_cf_model <- causal_forest(
  X = covars,                                                          ①
  Y = as.vector(nsw_data$re78),
  W = as.vector(nsw_data$treat),
  tune.parameters = "all",
  seed = 88)                                                           ②

  nsw_ite_predict <-predict(nsw_cf_model, estimate.variance = TRUE)    ③
  nsw_ite <- nsw_ite_predict$predictions                               ④
library(plotly)
```

```r
library(grf)
grid <- expand.grid(age = age_seq,educ = educ_seq)

mean_values <- colMeans(nsw_data, na.rm = TRUE)
other_vars <- names(nsw_data)[!(names(nsw_data) %in%  c("age", "educ",
                                    "response","ite", "re78","re75", "data_id","treat"))]

for (var in other_vars) {
grid[[var]] <- mean_values[var]
}

grid$predictions <- predict(nsw_cf_model, newdata = grid)

response_matrix <- matrix(as.numeric(unlist(grid$predictions)),
                          nrow = length(educ_seq), ncol = length(age_seq))

fig <- plot_ly(
x = ~age_seq,
y = ~educ_seq,
z = ~response_matrix,
type = 'surface',
colorscale = list(
c(0.25, 'rgb(0, 255, 255)')
),
showscale = FALSE
)
```

```
camera <- list(
eye = list(x=-1.7, y=-1.7, z=1.7),
center = list(x = 0, y = 0, z = 0),
up = list(x = 0, y = 0, z = 1)
)


fig <- fig %>% layout(
scene = list(
xaxis = list(title = 'Age'),
yaxis = list(title = 'Education'),
zaxis = list(title = "ITE in $'s"),
camera = camera
),
title = '3D Response Surface of Causal Forest: Age vs Education'
)
fig
# library(ggplot2)
# library(ggthemes)
# library(scales)
# ggplot(nsw_data) +
#   aes(x = as.factor(degree), y = ite) +
#   geom_boxplot()
# ggplot(nsw_data) +
#   aes(x = as.factor(marr), y = ite) +
#   geom_boxplot()
# ggplot(nsw_data) +
#   aes(x = as.factor(black), y = ite) +
#   geom_boxplot()
# ggplot(nsw_data) +
#   aes(x = as.factor(hisp), y = ite) +
#   geom_boxplot()
library(ggplot2)
library(ggthemes)
library(scales)
ggplot(nsw_data, aes(x = educ, y = nsw_ite)) +
  geom_point(size=2) +
  labs(
    title = "Visualising Treatment Effect Heterogeneity",
    subtitle = "Years of Education vs. Estimated Individual Treatment Effect",
    y = "Individual Treatment Effect ($ Increase in Earnings in 1978)",
    x = "Years of Education") +
  geom_rangeframe() +
  theme_tufte(base_size = 14)+
  theme(
    plot.title = element_text(size = 20),
    plot.subtitle = element_text(size = 15),
    legend.position = "bottom") +
  scale_x_continuous(breaks = pretty_breaks(n = 15)) +
  scale_y_continuous(breaks = pretty_breaks(n = 10))
library(ggplot2)
```

```r
library(ggthemes)

library(scales)

ggplot(nsw_data, aes(x = age, y = nsw_ite, color = as.factor(degree))) +

  geom_point(size=2) +

  labs(

    title = "Visualising Treatment Effect Heterogeneity",

    subtitle = "Age of Participant vs. Estimated Individual Treatment Effect",

    y = "Individual Treatment Effect ($ Increase in Earnings in 1978)",

    x = "Age (Years)") +

  geom_rangeframe() +

  theme_tufte(base_size = 14)+

  theme(

    plot.title = element_text(size = 20),

    plot.subtitle = element_text(size = 15),

    legend.position = "bottom") +

  scale_x_continuous(breaks = pretty_breaks(n = 15)) +

  scale_y_continuous(breaks = pretty_breaks(n = 10)) +

  scale_color_manual("Bachelor Degree Completion:",values = c("1" = "red",

                                                              "0" = "black" ),

                     labels = c("1" = "Degree","0" = "No Degree"))

save.image(file = "my_environment.RData")
```

6 Summary

7 Appendix

7.1 Datasets

7.1.1 NSW Jobs Dataset

The National Supported Work (NSW) Demonstration Job Training Program dataset originates from a large-scale social experiment conducted in the 1970s aimed at evaluating the impact of job training on employment and earnings among disadvantaged groups, including ex-addicts, ex-offenders, youth dropouts, and long-term unemployed women. The data contains a wide range of covariates including as age, education, pre-treatment earnings, marital status, and race.

The study is a randomized controlled trial (RCT) design which is rare for jobs and employment data. Participants were randomly assigned to either a treatment group, which received job training, or a control group, which did not. This randomization is notable as it as it simplifies the calculation of a treatment effect.

Initially (**lalonde1986?**) used the NSW dataset to compare experimental and non-experimental estimators of the treatment effect. His findings highlighted significant discrepancies between the two, underscoring the importance of randomization in estimating causal effects. This study has been widely cited and forms the basis for many discussions on the validity of non-experimental methods. Following this, (**dehejia1999?**), who revisited LaLonde's analysis and compared many different contemporary methods with varying results.

For these reason it is commonly used in the literature as a toy dataset and features in (**mixtape20?**), paper,paper, paper. It serves as a practical example for students learning about causal inference, allowing them to understand and apply different econometric methods.

```r
library(caret)
library('causaldata')
data("nsw_mixtape", package = "causaldata")
nsw_data <- as.data.frame(nsw_mixtape)

nsw_data$data_id <- seq(1,length(nsw_data$data_id))

nsw_data$degree <- abs(nsw_data$nodegree-1)

nsw_data$nodegree <- NULL

#nsw_split <- createDataPartition(nsw_data$data_id, p=0.5,list = FALSE)

#nsw_train <- nsw_data[nsw_split,]
#nsw_est <- nsw_data[-nsw_split,]
```

7.1.2 Coffee Dataset

blabla

```r
library(haven)
coffee_data <-  read_dta("C:/Users/mitch/OneDrive - University of Otago/Mitchell Research/Pro



coffee_data <- zap_formats(coffee_data)


#coffee_data <- na.omit(coffee_data)
#remeber to leave in or else cant predcit obs and shit goes wrong.
coffee_data <- coffee_data[-c(56,84,156 ),]
```

```r
library(readr)
liver_data <- read_csv("C:/Users/mitch/OneDrive - University of Otago/Mitchell Research/cirrh
```

```
Rows: 418 Columns: 20
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (7): Status, Drug, Sex, Ascites, Hepatomegaly, Spiders, Edema
dbl (13): ID, N_Days, Age, Bilirubin, Cholesterol, Albumin, Copper, Alk_Phos...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

7.2 Functions

```
extract_comparison_results <- function(results) {
  extracted_results <- data.frame(
    Estimate = results$estimate,
    SE = results$std.error,
    P.Value = results$p.value,
    Lower.CI = results$conf.low,
    Upper.CI = results$conf.high
  )

  return(extracted_results)
}
```

7.3 Theming

```
library(ggplot2)
library(showtext)
```

```
Loading required package: sysfonts
```

```
Loading required package: showtextdb
```

```r
font_add_google(name = "Source Sans Pro", family = "Source Sans Pro")

showtext_auto()

custom_ggplot_theme <- theme_classic(base_size = 11, base_family = "Source Sans Pro") +
    theme(
      text = element_text(color = "#333333"),
      plot.background = element_blank(),  # No plot background
      panel.background = element_blank(),  # No panel background
      axis.text = element_text(color = "#333333"),
      axis.title = element_text(color = "#333333", face = "bold"),
      legend.text = element_text(color = "#333333"),
      legend.title = element_text(color = "#333333", face = "bold",),
      plot.title = element_text(size = 14, face = "bold", color = "#333333"),
      plot.subtitle = element_text(size = 12, face = "italic", color = "#333333"),
      strip.text = element_text(face = "bold", family = "Source Sans Pro",
                                color = "#333333"),
      legend.position="bottom")

theme_set(custom_ggplot_theme)
```

7.4 Agnolagements

References

Andam, Kwaw S., Paul J. Ferraro, Alexander Pfaff, G. Arturo Sanchez-Azofeifa, and Juan A. Robalino. 2008. "Measuring the effectiveness of protected area networks in reducing deforestation." *Proceedings of the National Academy of Sciences* 105 (42): 16089–94. https://doi.org/10.1073/pnas.0800437105.

Athey, Susan, and Guido Imbens. 2016. "Recursive partitioning for heterogeneous causal effects." *Proceedings of the National Academy of Sciences of the United States of America* 113 (27): 7353–60. https://doi.org/10.1073/pnas.1510489113.

Athey, Susan, Julie Tibshirani, and Stefan Wager. 2019. "Generalized random forests." *Annals of Statistics* 47 (2): 1179–1203. https://doi.org/10.1214/18-AOS1709.

Austin, Peter C. 2008. "A critical appraisal of propensity-score matching in the medical literature between 1996 and 2003." *Statistics in Medicine* 27 (April): 2037–49. https://doi.org/10.1002/sim.3150.

Bader-El-Den, Mohammed, Eleman Teitei, and Todd Perry. 2019. "Biased Random Forest for Dealing with the Class Imbalance Problem." *IEEE Transactions on Neural Networks and Learning Systems* 30 (7): 2163–72. https://doi.org/10.1109/TNNLS.2018.2878400.

Breiman, Leo. 1996. "Bagging predictors." *Machine Learning* 24: 123–40. https://doi.org/10.3390/risks8030083.

Breiman, L, Jerome H Friedman, Richard A Olshen, and C J Stone. 1984. "Classification and Regression Trees." *Biometrics* 40: 874. https://api.semanticscholar.org/CorpusID:

29458883.

Brieman, Leo. 2001. "Random Forests." *Machine Learning* 45: 5–32. https://doi.org/https://doi.org/10.1023/A:1010933404324.

Brookhart, M. Alan, Sebastian Schneeweiss, Kenneth J. Rothman, Robert J. Glynn, Jerry Avorn, and Til Stürmer. 2006. "Variable selection for propensity score models." *American Journal of Epidemiology* 163 (12): 1149–56. https://doi.org/10.1093/aje/kwj149.

Cannas, Massimo, and Bruno Arpino. 2019. "A comparison of machine learning algorithms and covariate balance measures for propensity score matching and weighting." *Biometrical Journal* 61 (4): 1049–72. https://doi.org/10.1002/bimj.201800132.

Card, David, and Alan B Krueger. 1994. "Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania." *American Economic Review* 84 (4): 772–93. https://doi.org/10.3386/w4509.

Chu, Yu-Wei Luke, and Wilbur Townsend. 2019. "Joint culpability: The effects of medical marijuana laws on crime." *Journal of Economic Behavior and Organization* 159: 502–25. https://doi.org/10.1016/j.jebo.2018.07.003.

Cunningham, Scott. 2021a. "Directed Acyclic Graphs." In *Causal Inference: The Mixtape*, 96–118. Yale University Press. https://doi.org/10.2307/j.ctv1c29t27.6.

———. 2021b. "Matching and Subclassification." In *Causal Inference: The Mixtape*, 175–240. Yale University Press. https://doi.org/10.2307/j.ctv1c29t27.8.

Ferri-García, Ramón, and María Del Mar Rueda. 2020. "Propensity score adjustment using machine learning classification algorithms to control selection bias in online surveys." *PLoS ONE* 15 (4): 1–19. https://doi.org/10.1371/journal.pone.0231500.

Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics* 29 (5): 1189–1232. https://www.jstor.org/stable/2699986.

Goller, Daniel, Michael Lechner, Andreas Moczall, and Joachim Wolff. 2020. "Does the estimation of the propensity score by machine learning improve matching estimation? The case of Germany's programmes for long term unemployed." *Labour Economics* 65 (March). https://doi.org/10.1016/j.labeco.2020.101855.

Heinrich, Carolyn. 2010. "A Primer for Applying Propensity-Score Matching." *Development*, no. August: 59. http://www.iadb.org/document.cfm?id=35320229.

Jawadekar, Neal, Katrina Kezios, Michelle C. Odden, Jeanette A. Stingone, Sebastian Calonico, Kara Rudolph, and Adina Zeki Al Hazzouri. 2023. "Practical Guide to Honest Causal Forests for Identifying Heterogeneous Treatment Effects." *American Journal of Epidemiology* 192 (7): 1155–65. https://doi.org/10.1093/aje/kwad043.

King, Gary, and Richard Nielsen. 2019. "Why Propensity Scores Should Not Be Used for Matching." *Political Analysis* 27 (4): 435–54. https://doi.org/10.1017/pan.2019.11.

Kovach, Matthew. 2018. "Causal Inference of Human Resources Key Performance Indicators." PhD thesis, Bowling Green State University. https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/.

Künzel, Sören R., Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. 2019. "Metalearners for estimating heterogeneous treatment effects using machine learning." *Proceedings of the National Academy of Sciences of the United States of America* 116 (10): 4156–65. https://doi.org/10.1073/pnas.1804597116.

Lee, Brian K., Justin Lessler, and Elizabeth A. Stuart. 2010. "Improving propensity score weighting using machine learning." *Statistics in Medicine* 29: 337–46. https://doi.org/10.1002/sim.3782.

Lewbel, Arthur. 2007. "A local generalized method of moments estimator." *Economics Letters* 94 (1): 124–28. https://doi.org/10.1016/j.econlet.2006.08.011.

McCaffrey, Daniel F., Greg Ridgeway, and Andrew R. Morral. 2004. "Propensity score estimation with boosted regression for evaluating causal effects in observational studies." *Psychological Methods* 9 (4): 403–25. https://doi.org/10.1037/1082-989X.9.4.403.

Nabi, Razieh, Joel Pfeiffer, Denis Charles, and Emre Kıcıman. 2022. "Causal Inference in the

Presence of Interference in Sponsored Search Advertising." *Frontiers in Big Data* 5: 1–12. https://doi.org/10.3389/fdata.2022.888592.

Naimi, Ashley I., Stephen R. Cole, and Edward H. Kennedy. 2017. "An introduction to g methods." *International Journal of Epidemiology* 46 (2): 756–62. https://doi.org/10.1093/ije/dyw323.

Nie, X., and S. Wager. 2021. "Quasi-oracle estimation of heterogeneous treatment effects." *Biometrika* 108 (2): 299–319. https://doi.org/10.1093/biomet/asaa076.

Noroozi, Mehdi, Peter Higgs, Alireza Noroozi, Bahram Armoon, Bentolhoda Mousavi, Rosa Alikhani, Mohammad Rafi Bazrafshan, Ali Nazeri Astaneh, Azadeh Bayani, and Ladan Fattah Moghaddam. 2020. "Methamphetamine use and HIV risk behavior among men who inject drugs: Causal inference using coarsened exact matching." *Harm Reduction Journal* 17 (66). https://doi.org/10.1186/s12954-020-00411-1.

Olson, Matthew A., and Abraham J. Wyner. 2018. "Making Sense of Random Forest Probabilities: a Kernel Perspective," 1–35. http://arxiv.org/abs/1812.05792.

Pan, Yao, Stephen C. Smith, and Munshi Sulaiman. 2018. "Agricultural extension and technology adoption for food security: Evidence from Uganda." *American Journal of Agricultural Economics* 100 (4): 1012–31. https://doi.org/10.1093/ajae/aay012.

Ramachandra, Vikas. 2019. "Causal inference for climate change events from satellite image time series using computer vision and deep learning." http://arxiv.org/abs/1910.11492.

Ridgeway, Greg, Dan Mccaffrey, Andrew Morral, Matthew Cefalu, Lane Burgette, and Beth Ann Griffin. 2024. "Toolkit for Weighting and Analysis of Nonequivalent Groups: A Tutorial for the R TWANG Package." https://doi.org/10.7249/tl136.1.

Robinson, P.m. 1988. "Root-N-Consistent Semiparametric Regression." *The Econometric Socieity* 56 (4): 931–54. https://www.jstor.org/stable/1912705.

Rosenbaum, Paul R., and Donald B. Rubin. 1983. "The central role of the propensity score in observational studies for causal effects." *Biometrika* 70 (1): 41–55. https://doi.org/10.

1017/CBO9780511810725.016.

Rubin, Donald B. 1974. “Estimating Causal Effects of Treatments in Experimental and Observational Studies.” *Journal of Educational Psychology* 66 (5): 688–701. https://doi.org/10.1002/j.2333-8504.1972.tb00631.x.

Schuster, Tibor, Wilfrid Kouokam Lowe, and Robert W. Platt. 2016. “Propensity score model overfitting led to inflated variance of estimated odds ratios.” *Journal of Clinical Epidemiology* 80: 97–106. https://doi.org/10.1016/j.jclinepi.2016.05.017.

Schwab, Simon, and Leonhard Held. 2020. “Different worlds Confirmatory versus exploratory research.” *Significance* 17 (2): 8–9. https://doi.org/10.1111/1740-9713.01369.

Setoguchi, Soko, Sebastian Schneeweiss, Alan M. Brookhart, Robert J. Glynn, and Francis E. Cook. 2008. “Evaluating uses of data mining techniques in propensity score estimation: a simulation study.” *Pharmacoepidemiology and Drug Safety* 17 (March): 546–55. https://doi.org/10.1002/pds.

Smith, Jeffrey A., and Petra E. Todd. 2005. *Does matching overcome LaLonde’s critique of nonexperimental estimators?* Vol. 125. 1-2 SPEC. ISS. https://doi.org/10.1016/j.jeconom.2004.04.011.

Splawa-Neyman, Jerzy. 1923. “On the application of probability theory to agricultural experiments. Essay on principles. Section 9.” PhD thesis. https://doi.org/10.1214/ss/1177012031.

Tafforin, Carole, and Nancy L. Segal. 2022. “An experiment in cotwin control: Adaptation to space travel.” In *Twin Research for Everyone*, 617–24. Elsevier Inc. https://doi.org/https://doi.org/10.1016/C2019-0-02208-X.

Tibshirani, Robert. 1996. “Regression Shrinkage and Selection via the Lasso.” *Journal of the Royal Statistical Society* 58 (1): 267–88. https://www.jstor.org/stable/pdf/2346178.pdf?refreqid=fastly-default%7B/%%7D3Aff57285d6b8854126d21a135984fc4ca%7B/&%7Dab%7B/_%7Dsegments=%7B/&%7Dorigin=%7B/&%7Dinitiator=%7B/&%

7DacceptTC=1.

Tu, Chunhao. 2019. "Comparison of various machine learning algorithms for estimating generalized propensity score." *Journal of Statistical Computation and Simulation* 89 (4): 708–19. https://doi.org/10.1080/00949655.2019.1571059.

Wager, Stefan, and Susan Athey. 2018. "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests." *Journal of the American Statistical Association* 113 (523): 1228–42. https://doi.org/10.1080/01621459.2017.1319839.

Wager, Stefan, Trevor Hastie, and Bradley Efron. 2014. "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife." *Journal of Machine Learning Research* 15: 1625–51. https://arxiv.org/abs/1311.4555.

Weisberg, Sanford, and R D Cook. 1982. *Residuals and Influence in Regression.* Chapman & Hall.

Zhou, Xiaodan, Kevin Josey, Leila Kamareddine, Miah C. Caine, Tianjia Liu, Loretta J. Mickley, Matthew Cooper, and Francesca Dominici. 2021. "Excess of COVID-19 cases and deaths due to fine particulate matter exposure during the 2020 wildfires in the United States." *Science Advances* 7 (33). https://doi.org/10.1126/sciadv.abi8789.